SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 1-15, February 1989

# ANALYSIS OF AN INFINITE PRODUCT ALGORITHM*

J.-P. ALLOUCHE†, P. HAJNAL‡, AND J. O. SHALLIT§

**Abstract.** Let $w \in (0+1)^*$ be a finite nonempty string of zeros and ones, and let $a_w(n)$ denote the number of (possibly overlapping) occurrences of $w$ in the binary expansion of $n$.

Allouche and Shallit have recently shown that there exists an effectively computable rational function $b_w(n)$ such that

$$\sum_{n \geqq 0} \log_2 (b_w(n)) X^{a_w(n)} = \frac{1}{X-1}$$

for all complex $X$ such that $|X| \leqq 1$ and $X \neq 1$. They gave an algorithm to determine $b_w(n)$.

It is shown that the algorithm to determine $b_w(n)$ is related to a certain labeled binary tree $T(w)$. This observation allows two identities to be proven for the rational functions $b_w(n)$.

Combinatorial methods are used to investigate the structure of the tree $T(w)$. As the running time of the algorithm is proportional to the total number of nodes in the tree $T(w)$, the algorithm in this paper is shown to run in polynomial time by proving that $|T(w)| = O(|w|^{11.1})$. The existence of infinitely many strings $w$ such that $|T(w)| \geqq c|w|^3$ is also shown.

**Key words.** infinite product, binary string, polynomial-time algorithm

**AMS(MOS) subject classifications.** 68Q25, 11A63

## 1. Introduction.

Let $w \in (0+1)^*$ be a finite nonempty string of zeros and ones, and let $a_w(n)$ denote the number of (possibly overlapping) occurrences of $w$ in the binary expansion of $n$.

In [AS] Allouche and Shallit have shown there exists an effectively computable rational function $b_w(n)$ such that

$$\sum_{n \geqq 0} \log_2 (b_w(n)) X^{a_w(n)} = \frac{1}{X-1} \tag{1}$$

for all complex $X$ with $|X| \leqq 1$ and $X \neq 1$. If we set $w = 1$, $X = -1$, and exponentiate, we obtain the unusual infinite product of Woods and Robbins [Woo], [Rob]:

$$\prod_{n \geqq 0} \left(\frac{2n+1}{2n+2}\right)^{(-1)^{a_1(n)}} = \frac{\sqrt{2}}{2},$$

justifying the title of this paper. See also [ACMS], [Sha].

Lemmas 4 and 5 of [AS] assert that if we write $w = w_1 w_2 \cdots w_m$, then the function $b_w(n)$ is given by the following formula:

$$b_w(n) = U_w(w_1 w_2 \cdots w_{m-1}, w_m, n), \tag{2}$$

where $U_w$ is defined as follows.

* Received by the editors January 25, 1988; accepted for publication (in revised form) June 13, 1988.

† U. A. 226, Mathématiques et Informatique, Université de Bordeaux, 351 cours de la Libération, 33405 Talence Cedex, France.

‡ Department of Computer Science, University of Chicago, 1100 East 58th Street, Chicago, Illinois 60637. Present address, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

§ Department of Computer Science, University of Chicago, 1100 East 58th Street, Chicago, Illinois 60637. Present address, Department of Mathematics and Computer Science, Dartmouth College, Hanover, New Hampshire 03755.

Let $z = z_1 z_2 \cdots z_r$ and $t$ be strings. Then

$$(3)\qquad U_w(z,t,n) = \begin{cases} U_w(z_1 z_2 \cdots z_{r-1}, z_r t, n) & \text{if } r \geqq 1 \text{ and } z \text{ is not a suffix of } w, \\ \dfrac{U_w(z_2 z_3 \cdots z_r, t, n)}{U_w(\overline{z_1} z_2 \cdots z_{r-1}, z_r t, n)} & \text{if } r \geqq 2 \text{ and } z \text{ is a suffix of } w, \\ \dfrac{U_w(\varepsilon, t, n)}{U_w(\varepsilon, \overline{z_r} t, n)} & \text{if } r = 1 \text{ and } z \text{ is a suffix of } w, \\ \dfrac{2^{|t|} n + v(t)}{2^{|t|} n + v(t) + 1} & \text{if } r = 0. \end{cases}$$

Here $\bar{x}$ for a bit $x$ denotes $1 - x$, the *complement* of $x$; $\varepsilon$ denotes the empty string, $|t|$ denotes the number of symbols in the string $t$, and $v(t)$ denotes the value of $t$ when interpreted as the binary expansion of a nonnegative integer.

For example,

$$(4)\qquad b_{110110}(n) = U_{110110}(11011, 0, n) = \frac{8n+6}{8n+7} \cdot \frac{16n+15}{16n+14} \cdot \frac{16n+7}{16n+6} \cdot \frac{64n+54}{64n+55}.$$

The reader will find a table of $b_w(n)$ for all $w$ with $1 \leqq |w| \leqq 3$ in the Appendix.

Formulas (2) and (3) suggest a recursive algorithm for computing the rational function $b_w(n)$. In this paper, we are interested in the behavior of this algorithm. More precisely, we discuss the number of times formula (3) is invoked when computing $b_w(n)$.

The first observation necessary is that we can construct a binary tree out of the first arguments $z$ of the function $U_w$ that reflects the structure of the computation of $U_w(z, t, n)$, i.e., a computation tree. We do this as follows.

DEFINITIONS. Let $w$ and $z$ be strings and let $T_w(z)$ be a labeled binary tree defined recursively as follows.

The root of $T_w(z)$ is a node labeled $z = z_1 z_2 \cdots z_m$. If $z$ is not a suffix of $w$, then the root has one subtree given by $T_w(z_1 z_2 \cdots z_{m-1})$. If $z$ is a suffix of $w$, then the root has two subtrees: a left subtree given by $T_w(z_2 z_3 \cdots z_m)$, and a right subtree given by $T_w(\overline{z_1} z_2 \cdots z_{m-1})$. Such a node, with outdegree 2, is called a *branching* node. The *leaves* of the tree $T_w(z)$ are the nodes labeled with empty strings, i.e., strings of length zero.

Notice that the labels of nodes that are the same distance from the root have the same lengths. Thus we can speak of the *level* of a node, which is defined to be the length of the label associated with it.

Let $L_w(z)$ denote the number of leaves in the tree $T_w(z)$. Let $|T_w(z)|$ denote the total number of nodes in the tree $T_w(z)$. If $w = z$, we will frequently omit the subscript, so $L(w)$ means the same thing as $L_w(w)$, and $T(w)$ means the same as $T_w(w)$.

For example, Fig. 1 illustrates the tree $T_{110110}(11011)$. This tree has six leaves and so $L_{110110}(11011) = 6$. The total number of nodes is 14 and so $|T_{110110}(11011)| = 14$.

If $w = w_1 w_2 \cdots w_m$ is a string and $k \leqq m$, then define

$$\operatorname{drop}(w, k) = w_1 w_2 \cdots w_{m-k}.$$

With this notation, we have the following theorem.

THEOREM 1.1. *The degree $d_w$ of the numerator and denominator of the rational function $b_w(n)$ is bounded above by $L_w(\operatorname{drop}(w, 1))$. The total number of invocations of the function $U$ that occur in the recursive computation of $b_w(n)$ by* (2) *and* (3) *is* $|T_w(\operatorname{drop}(w, 1))|$.

*Proof.* The only remark we must make is that the degree $d_w$ of $b_w(n)$ is not necessarily equal to $L_w(\operatorname{drop}(w, 1))$, as cancellation may occur among the terms represented by the leaves of the tree $T_w(\operatorname{drop}(w, 1))$. Such cancellation actually occurs in practice; for

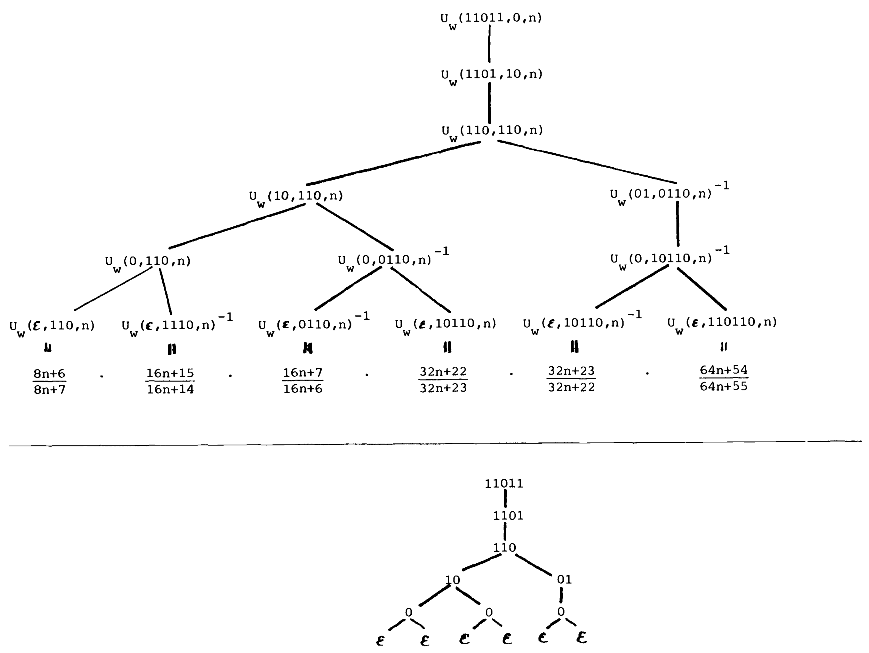


FIG. 1. *Computation of $b_w(n)$ and $T_w(\text{drop}\,(w,\,1))$ for $w$ = 110110.*

$w$ = 110110, we have $d_w = 4$, but as we have seen above, $L_w(\text{drop}\,(w,\,1)) = 6$. However, $L_w(\text{drop}\,(w,\,1))$ is certainly an *upper bound* for the degree. □

Now we state and prove two identities for the rational functions $b_w(n)$.

THEOREM 1.2.

$$b_w(n) = b_{0w}(n)b_{1w}(n), \tag{5}$$

$$b_{\bar{w}}(n) = \frac{1}{b_w(-n-1)}. \tag{6}$$

*Proof*. To prove (5), put $w = w_1w_2\cdots w_m$, and define $k$ so that $w_1w_2\cdots w_k$ is the longest prefix of $w_1w_2\cdots w_{m-1}$ that is also a suffix of $w$. Define the bit $x$ so that $xw_1w_2\cdots w_k$ is also a suffix of $w$. Then we find

$$\begin{aligned}
b_{xw}(n) &= U_{xw}(xw_1w_2\cdots w_{m-1}, w_m, n)\\
&= U_{xw}(xw_1w_2\cdots w_k, w_{k+1}\cdots w_m, n)\\
&= \frac{U_{xw}(w_1w_2\cdots w_k, w_{k+1}\cdots w_m, n)}{U_{xw}(\bar{x}w_1w_2\cdots w_{k-1}, w_k\cdots w_m, n)}\\
&= \frac{U_w(w_1w_2\cdots w_k, w_{k+1}\cdots w_m, n)}{U_{\bar{x}w}(\bar{x}w_1w_2\cdots w_{k-1}, w_k\cdots w_m, n)}\\
&= \frac{U_w(w_1w_2\cdots w_{m-1}, w_m, n)}{U_{\bar{x}w}(\bar{x}w_1w_2\cdots w_{m-1}, w_m, n)}\\
&= \frac{b_w(n)}{b_{\bar{x}w}(n)},
\end{aligned}$$

which proves the first result.

Now let us prove (6). Since $\bar{a}$ is a suffix of $\bar{w}$ precisely when $a$ is a suffix of $w$, it suffices to prove the second identity for $U_w(\varepsilon, t, n)$. Then we have

$$\begin{aligned} U_{\bar{w}}(\varepsilon,\bar{t},n) &= \frac{2^{|t|}n+v(\bar{t})}{2^{|t|}n+v(\bar{t})+1} \\ &= \frac{2^{|t|}n+2^{|t|}-1-v(t)}{2^{|t|}n+2^{|t|}-v(t)} \\ &= \frac{2^{|t|}(-n-1)+v(t)+1}{2^{|t|}(-n-1)+v(t)} \\ &= U_w(\varepsilon,t,-n-1)^{-1}, \end{aligned}$$

which proves the result. □

The remainder of this paper is devoted to discussing the asymptotic behavior of the functions $L_w(z)$ and $|T_w(z)|$. Since each node in the tree $T_w(z)$ lies on some path from the root to a leaf, it is clear that

(7) $$|T_w(z)| \leqq (1+|z|)L_w(z).$$

Let us define

(8) $$R(x) = \max_{|w| \leqq x} |T_w(\text{drop}(w,1))|.$$

We will show

(9) $$\frac{2}{243}x^3 \leqq R(x) \leqq 2x^{11.1}.$$

In particular, the upper bounds show that the algorithm given by (2) and (3) for computing $b_w(n)$ is actually a polynomial time algorithm.

We note that there is a relationship between our binary tree $T_w(\text{drop}(w, 1))$ and the "bifix-free" strings of Nielsen [Nie]. A string $w$ is said to be bifix-free if no proper prefix of $w$ is also a suffix of $w$. We can easily derive a recurrence for the number $B_n$ of bifix-free words of length $n$, and Nielsen has shown that

$$\lim_{n\to\infty} \frac{B_n}{2^n} = .267786\cdots.$$

Now if $w$ is bifix-free, it is easily seen that $L_w(\text{drop}(w, 1)) = 1$. In fact, in this case

$$b_w(n) = \frac{2^{|w|}+v(w)}{2^{|w|}+v(w)+1}.$$

It is also easily seen that $|T_w(\text{drop}(w, 1))| = |w|$. Thus for at least 26.7 percent of all $w$, the algorithm actually runs in *linear* time.

**2. The lower bound.** In this section, we obtain a lower bound for the function $R(x)$. We do this by explicitly constructing a set of strings for which the algorithm behaves badly.

We start with a series of four lemmas.

LEMMA 2.1. *For $k \geqq 0$ we have $|T(0^k)| = \frac{1}{2}(k+1)(k+2)$.*

*Proof.* Consider the tree $T_{0^k}(0^k)$ with root labeled $0^k$. Its left child is labeled $0^{k-1}$ and its right child is labeled $10^{k-2}$. No child of the node labeled $10^{k-2}$ can be a suffix of $0^k$, since it begins with a one. Hence,

$$|T(0^k)| = |T(0^{k-1})| + k + 1$$

for $k \geqq 1$. The result follows since $|T(\varepsilon)| = 1$. $\square$

COROLLARY. $|T(10^k)| = (k+1)(k+2)+1$.

*Proof.* The tree $T_{10^k}(10^k)$ consists of three pieces: a root labeled $10^k$; a left subtree with root labeled $0^k$; and a right subtree with root labeled $0^k$. Hence, $|T(10^k)| = 2|T(0^k)| + 1$, from which the result follows. $\square$

LEMMA 2.2. *For* $k \geqq 0$ *we have* $|T(0^k10^k)| = \frac{1}{2}(k+1)(5k+6)$.

*Proof.* Consider the tree with root $0^k10^k$, as portrayed in Fig. 2.

It is clear that none of the nodes on levels $2k$ through $k+1$ that are right children can themselves be branching nodes, since each of their labels contains two ones, and thus cannot be a suffix of $0^k10^k$. On level $k+1$ there are $k+1$ nodes, and the leftmost node is labeled $10^k$. None of the labels of descendents of the $k$ rightmost nodes can be a suffix of $0^k10^k$, since each such label begins with a 1 followed by at most $k-1$ zeros. Thus we see

$$|T(0^k10^k)| = |T(10^k)| + \sum_{i=k+3}^{2k+2} i$$

$$= (k+1)(k+2)+1+\tfrac{1}{2}(2k+2)(2k+3)-\tfrac{1}{2}(k+2)(k+3),$$

from which the result follows. $\square$

LEMMA 2.3. *Let* $j \geqq k \geqq 0$. *Then*

$$|T(0^j10^k)| = (j-1)k^2+(3j+2)k-k^3+\tfrac{1}{2}(j+1)(j+6).$$

*Proof.* Consider the tree with root labeled $0^j10^k$, as in Fig. 3.

Any node that is a right child must have a label of the form $10^{j-a}10^{k-b}$ for $a \geqq 1$. A prefix of this string can be a suffix of $0^j10^k$ provided $j-a \geqq k$, i.e., provided $a \leqq j-k$. Thus for each $a$, $1 \leqq a \leqq j-k$, we see that a descendent of a right child is of the form $10^k$ and so

$$|T(0^j10^k)| = (j-k)|T(10^k)| + \left(\sum_{i=k+1}^{j} i\right) + |T(0^k10^k)|$$

$$= (j-k)(1+(k+1)(k+2)) + \frac{1}{2}(k+1)(5k+6) + \frac{1}{2}j(j+1) - \frac{1}{2}k(k+1)$$

$$= (j-1)k^2+(3j+2)k-k^3+\frac{1}{2}(j+1)(j+6),$$

which is the desired result. $\square$

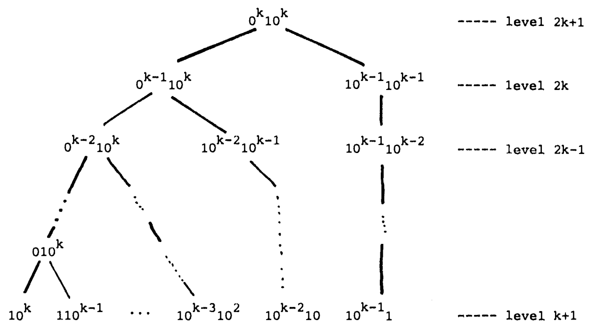


FIG. 2. $T(0^k10^k)$.

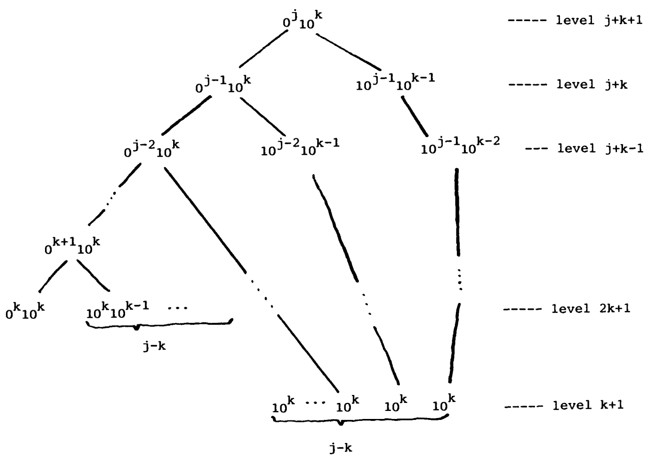


FIG. 3. *The tree* $T(0^j10^k)$.

LEMMA 2.4. *Let* $w = 0^j10^j10^k$ *where* $j \geqq k \geqq 0$. *Then*

$$|T_w(\text{drop}(w,1))| = |T_{0^j10^k}(0^j10^k)| + j.$$

*Proof.* The tree $T_w(\text{drop}(w, 1))$ consists of a root labeled $0^j10^j10^{k-1}$. This node, whose label is not a suffix of $0^j10^j10^k$, has only one child whose label is $0^j10^j10^{k-2}$. Continuing down the tree, it is easily seen that the levels $2j+k+1$ through $j+k+2$ of this tree each consist of nonbranching nodes. However, level $j+k+1$ consists of the node labeled $0^j10^k$, which is a suffix of $0^j10^j10^k$. From this, the stated result easily follows. □

We are now ready for the proof of the lower bound. Recall the definition of $R(x)$ given in (8) at the end of § 1.

THEOREM 2.5. *For all integers* $x \geqq 0$ *we have*

$$R(x) \geqq \frac{2x^3}{243} + \frac{17x^2}{72}.$$

*Proof.* Let $w = 0^j10^j10^k$. From Lemmas 2.3 and 2.4 we have

$$|T_w(\text{drop}(w,1))| = g(j,k),$$

where

$$g(j,k) = (j-1)k^2 + (3j+2)k - k^3 + \frac{j^2}{2} + \frac{9j}{2} + 3.$$

We wish to maximize $g(j, k)$ subject to $|w| = x = 2j+k+2$. We find

$$G(j) = g(j, x-2j-2)$$
$$= 12j^3 + \left(\frac{45}{2} - 16x\right)j^2 + \left(7x^2 - 21x + \frac{29}{2}\right)j - x^3 + 5x^2 - 6x + 3$$

and

$$\frac{dG}{dj} = 36j^2 + (45-32x)j + 7x^2 - 21x + \frac{29}{2}.$$

The solutions of $dG/dj = 0$ are given by

$$j = \frac{32x - 45 \pm \sqrt{16x^2 + 144x - 63}}{72}.$$

The maximum of $G(j)$ occurs when the negative sign is chosen; in this case we see that $j \approx (7/18)x$. Thus $g(j, k)$ is maximized when (roughly speaking) $j = 7b$ and $k = 4b$.

We now argue by cases, based on $x \pmod{18}$. For each of the 18 residue classes $x = 18b + a$, $0 \leqq a \leqq 17$, we find appropriate constants $p$ and $q$ such that $a = 2p + q + 2$ and $g(7b + p, 4b + q)$ is as large as possible. For example, for $a = 13$ we choose $p = 4$, $q = 3$ and find that $x = 18b + 13$ and

$$g(7b+4, 4b+3) = \frac{2x^3}{243} + \frac{17x^2}{72} + \frac{179x}{162} - \frac{2623}{1944}.$$

In fact, the reader can easily verify that for each $a$ there exist choices $p$ and $q$ such that

$$g(7b+p, 4b+q) = \frac{2x^3}{243} + \frac{17x^2}{72} + d(x),$$

where $d(x)$ is a linear function that satisfies $d(x) > 0$ for $x \geqq a$. Thus we have shown $R(x) > (2x^3/243) + (17x^2/72)$. $\square$

Let us define

$$S(x) = \max_{|w| \leqq x} L_w(\text{drop}(w, 1)).$$

The methods of this section can also be used to prove that

$$S(x) \geqq \frac{x^2}{12} + \frac{2x}{3} - \frac{8}{3}.$$

Thus there exist computation trees with a quadratic number of leaves. This bound does not lead to a lower bound on the worst-case of the degree $d_w$ of the rational function $b_w(n)$, however, as it does not take into account cancellation among the terms represented by the leaves.

In fact, we conjecture that $d_w \leqq |w|$, except when $w = (10)^n$, $(01)^n$, $(10)^n1$, or $(01)^n0$, where we have $d_w = \lfloor 3|w|/2 \rfloor - 2$.

**3. The upper bound: notation and outline of the proof.** In this section and the three which follow, we prove a polynomial upper bound for the function $L_w(w)$, which implies a polynomial upper bound for the function $R(x)$.

Observe that the tree $T_w(\text{drop}(w, 1))$ consists of (a) a simple path starting at the root and (b) a $T_u(u)$ tree rooted at the first branching point of this path. Then clearly $L_w(\text{drop}(w, 1)) = L_u(u)$, where $|u| < |w|$. Hence it suffices to give an upper bound on the number of leaves of a tree $T_u(u)$.

By abuse of notation, we will define the function $L$ with integer arguments by setting $L(n) = \max_{|u|=n} L_u(u)$. Our goal is to give a polynomial upper bound on $L(n)$. We can extend this function to the real numbers by defining $L(x) = \max_{|u| \leqq x} L_u(u)$.

We now define some symbols associated with the tree $T_w(w)$. There is a path starting at the root and going to the left until it reaches a leaf. At each node on this path, there is a branching, and the right subtree starts with a simple path until another branching occurs. Then this node is labeled with $y_i = \overline{w_i} w_{i+1} \cdots w_{a_i}$, which by definition is necessarily a suffix of $w$. We define $l_i = |y_i| = a_i - i + 1$. See Fig. 4.

We identify substrings of $w$ with intervals with endpoints in $U = \{1, 2, \cdots, n\}$. Let $I = [k, l]$ be an interval. Then we call $l$ the *final point*, $k$ the *initial point*, and the *length* of $I$ is $l - k + 1$. The *distance* between two points $s$ and $t$ in $U$ is $|s - t|$. The distance between two intervals is defined as the distance between their final points.

When we speak about the interval $I_i = [i, a_i]$, we should not forget that there is an underlying substring

$$w_i w_{i+1} \cdots w_{a_i}.$$

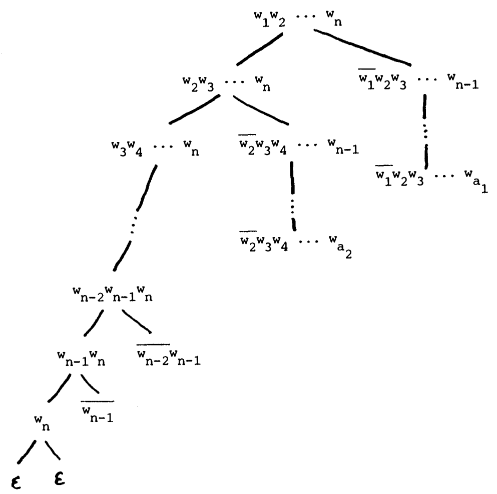

$w_1w_2\cdots w_n$
$w_2w_3\cdots w_n$
$\overline{w_1}w_2w_3\cdots w_{n-1}$
$w_3w_4\cdots w_n$
$\overline{w_2}w_3w_4\cdots w_{n-1}$
$\overline{w_1}w_2w_3\cdots w_{a_1}$
$\overline{w_2}w_3w_4\cdots w_{a_2}$
$w_{n-2}w_{n-1}w_n$
$w_{n-1}w_n$
$\overline{w_{n-2}w_{n-1}}$
$w_n$
$\overline{w_{n-1}}$
$\varepsilon$
$\varepsilon$

FIG. 4. *The tree $T(w)$.*

We refer to this corresponding substring as the string *spanned* by the interval. It will sometimes be useful to speak about the interval $I_i^*$, which omits the initial point of $I_i$. The string

$$w_{i+1}w_{i+2}\cdots w_{a_i}$$

spanned by the truncated interval $I_i^*$ will be denoted by $y_i^*$.

*Outline of the proof*. Choose a string $w$ of length $n$ such that $L(w) = L(n)$. This $w$ defines a tree $T_w(w)$. Then

$$L(n)=L(w)=1+\sum_{i=1}^{n} L(y_i)\leqq 1+\sum_{i=1}^{n} L(l_i). \tag{10}$$

The basic idea is that our interval system $\{I_i\}_{i=1}^n$ is not arbitrary. We know that each truncated interval spans a suffix of the original string. This implies that two intervals of similar length span almost the same string. If the two intervals are close together, then the spanned strings are almost periodic. This observation gives a lot of information about the corresponding subtree, and we will discuss this case in § 4. If this is not the case, and intervals of similar length are far apart, then we can conclude some information about the distribution of $\{l_i\}_{i=1}^n$ (§ 5). This gives a recursive inequality for $L(n)$. The solution of this inequality is given in § 6.

We call an interval $I_i = [i, a_i]$ *quasiperiodic* if the string

$$y_i^* = w_{i+1}\cdots w_{a_i}$$

can be written as $\beta\alpha^k$, where $\alpha$, $\beta$ are strings, $|\alpha| \leqq l_i/c$ and $\beta$ is a suffix of $\alpha$. Here $c > 2$ is a constant whose exact value will be specified in § 6; the reader is advised to keep in mind the case $c = 4$. The period $p = |\alpha|$ is chosen to be as small as possible.

We say that an interval $I_i$ is *aperiodic* if it is not quasiperiodic. We let $\mathscr{Q}$ be the set of quasiperiodic intervals, and $\mathscr{A}$ be the set of aperiodic intervals.

According to this classification of intervals we can divide the sum in (10) into two sums:

$$L(n)=1+\sum_{i=1}^{n} L(y_i)=1+\sum_{\substack{i\\ I_i\in\mathscr{Q}}} L(y_i)+\sum_{\substack{j\\ I_j\in\mathscr{A}}} L(y_j). \tag{11}$$

We will give upper bounds separately for the two cases.

## 4. Quasiperiodic intervals.

LEMMA 4.1. *Let $I_i$ be a quasiperiodic interval of period $p$, and let the associated string be*

$$\begin{aligned} y_i &= \overline{w_i} w_{i+1} \cdots w_{a_i} \\ &= a\alpha_j \alpha_{j+1} \cdots \alpha_{p-1} (\alpha_0 \alpha_1 \cdots \alpha_{p-1})^k. \end{aligned}$$

*Then*

$$L(y_i) \leqq 2 \frac{n}{p} L(2p).$$

*Proof.* The proof has three steps, which we outline as follows:

(a) First, each of the labels on level $p$ of the tree $T(y_i)$ is contained in the set

$$S = \{\alpha_0\alpha_1 \cdots \alpha_{p-1}, \overline{\alpha_0}\alpha_1 \cdots \alpha_{p-1}, \overline{\alpha_1}\alpha_2 \cdots \alpha_{p-1}\alpha_0, \cdots, \overline{\alpha_{p-1}}\alpha_0 \cdots \alpha_{p-2}\},$$

and no label occurs more than $2k - 1$ times;

(b) Next, $L(\alpha\alpha) = \sum_{\gamma \in S} L(\gamma)$;

(c) Finally, we conclude from (a) and (b) that

$$L(y_i) \leqq (2k-1)L(\alpha\alpha) \leqq 2kL(2p) \leqq 2\frac{n}{p}L(2p).$$

Let us begin with part (b). From the tree $T(\alpha\alpha)$ we see that

$$L(\alpha\alpha) = L(\alpha) + \sum_{j=0}^{p-1} L(\overline{\alpha_j}\alpha_{j+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{p-2}).$$

Now we claim that there is a simple path with no branching nodes from

$$\overline{\alpha_j}\alpha_{j+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{p-2} \quad \text{to} \quad \overline{\alpha_j}\alpha_{j+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{j-1}.$$

Suppose, to the contrary, that there was a branching node, which would have the label

$$v = \overline{\alpha_i}\alpha_{i+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{p-j}.$$

Since this is a branching node, $v$ must be a suffix of $\alpha\alpha$. Looking only at the last $p$ bits of the two strings, we see that

$$\alpha_{p-j+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{p-j} = \alpha_0 \cdots \alpha_{p-1}.$$

This implies that $\alpha$ is of period $\gcd(p, j) < p$, which is a contradiction. This completes the proof of part (b).

Part (a) is proved similarly. The tree $T(y_i)$ consists of the root, labeled $y_i$, and two subtrees. The left subtree is

$$T(\alpha_j \cdots \alpha_{p-1}\alpha^k)$$

and the right subtree is

$$T_{y_i}(\bar{a}\alpha_j \cdots \alpha_{p-1}\alpha^{k-1}\alpha_0 \cdots \alpha_{p-2}).$$

Let us consider the left subtree first. A periodicity argument similar to that used in the previous paragraph shows that the labels on level $p$ of this tree are (a) $\alpha_0 \cdots \alpha_{p-1}$ and (b) at most $k$ copies each of

$$\overline{\alpha_0}\alpha_1 \cdots \alpha_{p-1}, \overline{\alpha_1}\alpha_2 \cdots \alpha_{p-1}\alpha_0, \cdots, \overline{\alpha_{p-1}}\alpha_0 \cdots \alpha_{p-2}.$$

See Fig. 5.

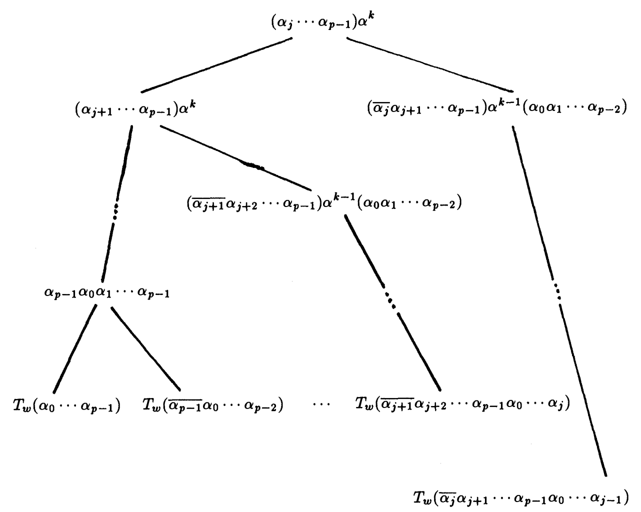

$(\alpha_j \cdots \alpha_{p-1})\alpha^k$
$(\alpha_{j+1} \cdots \alpha_{p-1})\alpha^k$
$(\overline{\alpha_j}\alpha_{j+1} \cdots \alpha_{p-1})\alpha^{k-1}(\alpha_0\alpha_1 \cdots \alpha_{p-2})$
$(\overline{\alpha_{j+1}}\alpha_{j+2} \cdots \alpha_{p-1})\alpha^{k-1}(\alpha_0\alpha_1 \cdots \alpha_{p-2})$
$\alpha_{p-1}\alpha_0\alpha_1 \cdots \alpha_{p-1}$
$T_w(\alpha_0 \cdots \alpha_{p-1})$
$T_w(\overline{\alpha_{p-1}}\alpha_0 \cdots \alpha_{p-2})$
$\cdots$
$T_w(\overline{\alpha_{j+1}}\alpha_{j+2} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_j)$
$T_w(\overline{\alpha_j}\alpha_{j+1} \cdots \alpha_{p-1}\alpha_0 \cdots \alpha_{j-1})$

FIG. 5. *The tree $T_w(w)$, where $w = (\alpha_j\alpha_{j+1} \cdots \alpha_{p-1}\alpha^k)$ $(w = \alpha_j\alpha_{j+1} \cdots \alpha_{p-1}\alpha^k)$.*

Now let us consider the right subtree. Again, a simple periodicity argument shows there is a simple path with no branching nodes from

$$v_1 = \bar{a}\alpha_j \cdots \alpha_{p-1}\alpha^{k-1}\alpha_0 \cdots \alpha_{p-2} \quad \text{to} \quad v_2 = \bar{a}\alpha_j \cdots \alpha_{p-1}\alpha^{k-1}.$$

This last node labeled $v_2$ is a branching node if and only if $\bar{a} = \alpha_{j-1}$. If it is not a branching node, then there is a simple path from

$$v_2 \quad \text{to} \quad v_3 = \overline{\alpha_{j-1}}\alpha_j \cdots \alpha_{p-1}\alpha_0\alpha_1 \cdots \alpha_{j-2}.$$

If $v_2$ is a branching node, then the left child of $v_2$ is labeled

$$\alpha_j \cdots \alpha_{p-1}\alpha^{k-1}$$

and the right subtree of $v_2$ is a simple path down to a node labeled $\alpha_0\alpha_1 \cdots \alpha_{p-1}$. Figure 5 again shows that there are at most $k - 1$ copies each of the labels

$$\overline{\alpha_0}\alpha_1 \cdots \alpha_{p-1}, \overline{\alpha_1}\alpha_2 \cdots \alpha_{p-1}\alpha_0, \cdots, \overline{\alpha_{p-1}}\alpha_0 \cdots \alpha_{p-2}$$

on level $p$ of the tree $T(v_4)$. Adding up all these possibilities gives the desired result. □

Lemma 4.1 shows that the upper bound in the case of quasiperiodic intervals depends on the size of the period. This suggests a partition of the quasiperiodic intervals according to their period size. Let us define

$$\mathcal{Q}_p = \{I : I \in \mathcal{Q} \text{ and its period is } p\}.$$

We will prove that $\mathcal{Q}_p$ cannot contain too many intervals, and most of these sets are empty. This gives a good upper bound.

LEMMA 4.2. *Let $I$ and $J$ be two arbitrary intervals, and suppose one of them spans a suffix of the string spanned by the other. If their distance is $d$, then one of the intervals spans a string of period $p$ with $p \leqq d$.*

*Proof.* The proof is clear. □

LEMMA 4.3.

$$\sum_{\substack{j \\ I_j \in \mathcal{Q}_p}} L(y_j) \leqq 2\frac{n^2}{p^2}L(2p).$$

*Proof.* From Lemma 4.2 we see that if two intervals are of period $p$, then they must be at least distance $p$ apart. Hence there can be at most $n/p$ intervals in $\mathcal{Q}_p$. The result follows when we apply Lemma 4.1.

To complete our upper bound on the quasiperiodic intervals, we need the following lemma.

LEMMA 4.4. *If $\mathcal{Q}_p$ and $\mathcal{Q}_q$ are nonempty and $p < q$, then $q \geqq (c-1)p$.*

*Proof.* Let $I_i \in \mathcal{Q}_p$ and $I_j \in \mathcal{Q}_q$. Suppose $p < q < (c-1)p$. Then $I_i^*$ spans a string $y_i^*$ such that

$$y_i^* = \beta\alpha^k = \cdots \alpha_r \alpha_{r+1} \cdots \alpha_{p-1}\alpha_0\alpha_1 \cdots \alpha_{p-1}\alpha_0\alpha_1 \cdots \alpha_{p-1} \tag{12}$$

and $I_j^*$ spans a string $y_j^*$ such that

$$y_j^* = \gamma\delta^{k'} = \cdots \delta_s\delta_{s+1} \cdots \delta_{q-1}\delta_0\delta_1 \cdots \delta_{q-1}. \tag{13}$$

Now both $y_i^*$ and $y_j^*$ match the label of the root, $w$, so the corresponding symbols in (12) and (13) must be equal. For each $\delta_h$ in the last period of $y_j^*$ there exists a corresponding $\alpha_l$ such that $\delta_h = \alpha_l$. The previous occurrence of this $\alpha_l$ has its own corresponding $\delta_m$. (To see that this $\delta_m$ actually exists, note that the construction fails only when $p + q \geqq l_i$. But $l_i \geqq cp$ by definition of quasiperiodic intervals, so $p + q \geqq cp$, and hence $q \geqq (c-1)p$, contrary to assumption.) Thus we find that for $0 \leqq h \leqq p$ we have $\delta_h = \delta_{h-p}$, where the subscripts are taken $(\operatorname{mod} q)$. This shows that $\delta = \delta_0\delta_1 \cdots \delta_{q-1}$ is actually periodic with period smaller than $\gcd(p, q) < q$, which is a contradiction. □

We now summarize the results of this section in an inequality.

LEMMA 4.5.

$$\sum_{\substack{i \\ I_i \in \mathcal{Q}}} L(y_i) \leqq \sum_{0 \leqq k \leqq \lfloor \log_{(c-1)} n \rfloor - 1} 2c^2(c-1)^{2k+2} L\left(\frac{2n}{c(c-1)^k}\right). \tag{14}$$

*Proof.* Lemma 4.4 shows that each interval of the form

$$\left(\frac{n}{c(c-1)^{k+1}}, \frac{n}{c(c-1)^k}\right],$$

$k = 0, 1, \cdots, \lfloor \log_{(c-1)} n \rfloor - 1$ contains at most one period $p$ for which $\mathcal{Q}_p$ is nonempty. Thus

$$\begin{aligned}
\sum_{\substack{i \\ I_i \in \mathcal{Q}}} L(y_i) &\leqq \sum_{\substack{p \\ \mathcal{Q}_p \neq \varnothing}} \sum_{\substack{i \\ I_i \in \mathcal{Q}_p}} L(y_i) \\
&\leqq \sum_{\substack{p \\ \mathcal{Q}_p \neq \varnothing}} 2\frac{n^2}{p^2} L(2p) \\
&\leqq \sum_{0 \leqq k \leqq \lfloor \log_{(c-1)} n \rfloor - 1} 2c^2(c-1)^{2k+2} L\left(\frac{2n}{c(c-1)^k}\right).
\end{aligned}$$

□

**5. Aperiodic intervals.** Recall that $\mathcal{A}$ is the set of aperiodic intervals. We divide $\mathcal{A}$ into classes depending on the lengths of the intervals. For $i \geqq 1$ define

$$\mathcal{A}_i = \left\{ I_j \in \mathcal{A} \text{ and } n \Big/ \left(\frac{c+1}{c}\right)^{i+1} \leqq l_j < n \Big/ \left(\frac{c+1}{c}\right)^i \right\}.$$

If an aperiodic interval $I_j$ belongs to $\mathcal{A}_i$ then we estimate $L(y_j)$ with $L(((c+1)/c)^i)$. To be sure that in estimating $L(n)$ we do not use the same function with a larger argument, we need the following lemma.

LEMMA 5.1. *If $l_j \geqq (c/(c+1))n$, then $I_j$ is quasiperiodic.*

*Proof.* $I_j$ spans a string $y_j$ that is a suffix of the word $w$. The distance $d$ between $I_j$ and the interval $[1, n]$ satisfies $d \leqq n - l_j \leqq n/(c+1) \leqq l_j/c$. From Lemma 4.5 it follows that $I_j$ spans an interval of period $p \leqq d \leqq l_j/c$. Hence $I_j$ is quasiperiodic. □

The most important lemma of this section is the following, which says that the elements of $\mathcal{A}_i$ are far apart. In particular, this implies that $\mathcal{A}_i$ contains only a few elements.

LEMMA 5.2. *If $I_k$ and $I_l$ are intervals from $\mathcal{A}_i$ $(i \geqq 1)$, then their distance is at least*

$$\frac{1}{c}\cdot n\Big/\left(\frac{c+1}{c}\right)^{i+1}.$$

*Proof.* We can consider the two truncated intervals $I_k^*$ and $I_l^*$ and their associated strings $y_k^*$ and $y_l^*$. The two truncated intervals have the same distance as the original ones, let us say $d$. One of $y_k^*$ and $y_l^*$ is a suffix of the other, so from Lemma 4.5 one of them has period at most $d$. Because it is an aperiodic interval, the length of this period $p$ satisfies $p > l_i/c$. Since $p \leqq d$, we find

$$d \geqq \frac{1}{c}\cdot n\Big/\left(\frac{c+1}{c}\right)^{i+1}. \qquad \square$$

COROLLARY 5.3.

$$|\mathcal{A}_i| \leqq \frac{n}{d} \leqq c\left(\frac{c+1}{c}\right)^{i+1} = (c+1)\left(\frac{c+1}{c}\right)^{i}.$$

We can now summarize the results of this section in the following inequality.

LEMMA 5.4.

$$\sum_{\substack{i \\ I_i \in \mathcal{A}}} L(w_i) \leqq \sum_{1 \leqq j \leqq \log_{(c+1)/c} n} (c+1)\left(\frac{c+1}{c}\right)^{j} L\left(n\Big/\left(\frac{c+1}{c}\right)^{j}\right).$$

*Proof.*

$$\sum_{\substack{i \\ I_i \in \mathcal{A}}} L(w_i) \leqq \sum_{1 \leqq j \leqq \log_{(c+1)/c} n} |\mathcal{A}_j|\, L\left(n\Big/\left(\frac{c+1}{c}\right)^{j}\right)$$

$$\leqq \sum_{1 \leqq j \leqq \log_{(c+1)/c} n} (c+1)\left(\frac{c+1}{c}\right)^{j} L\left(n\Big/\left(\frac{c+1}{c}\right)^{j}\right).$$

**6. Completing the upper bound proof.** To complete the proof of the upper bound we need to summarize our results and solve the resulting inequality.

By combining the estimates of the previous two sections, we see that

$$L(n) \leqq 1 + 2c^2 \sum_{0 \leqq k \leqq \log_{(c-1)} n} (c-1)^{2k+2} L\left(\frac{2n}{c(c-1)^k}\right)$$
$$+ (c+1) \sum_{1 \leqq j \leqq \log_{(c+1)/c} n} \left(\frac{c+1}{c}\right)^{j} L\left(n\Big/\left(\frac{c+1}{c}\right)^{j}\right).$$

We now prove that $L(n) \leqq 2n^{10.1}$, by induction on $n$. We assume that $L(n) \leqq Dn^a$ for all $n$ sufficiently small, and we determine $D$ and $a$ later. Now assume it is true for all $n < N$, and we wish to prove the assertion for $N$.

It suffices to show that

$$1 + DN^a\Bigg(2^{a+1}c^{2-a}(c-1)^2 \sum_{0 \leqq k \leqq \log_{(c-1)} n} (c-1)^{k(2-a)} + (c+1) \sum_{1 \leqq j \leqq \log_{(c+1)/c} n} \left(\frac{c+1}{c}\right)^{j(1-a)}\Bigg) \leqq DN^a.$$

We bound the first sum by

$$2^{a+1}c^{2-a}(c-1)^2 \sum_{k \geqq 0} (c-1)^{k(2-a)} = \frac{2^{a+1}c^{2-a}(c-1)^2}{1-(c-1)^{2-a}},$$

and the second sum by

$$(c+1) \sum_{j \leqq 1} \left(\frac{c+1}{c}\right)^{j(1-a)} = (c+1)\left(\left(\frac{c+1}{c}\right)^{1-a} \Bigg/ \left(1 - \left(\frac{c+1}{c}\right)^{1-a}\right)\right).$$

If we now write

$$\varphi_c(a) = \frac{2^{a+1}c^{2-a}(c-1)^2}{1-(c-1)^{2-a}} + (c+1)\left(\left(\frac{c+1}{c}\right)^{1-a} \Bigg/ \left(1 - \left(\frac{c+1}{c}\right)^{1-a}\right)\right),$$

then we wish to find $a$, $D$ such that

$$1 + DN^a \varphi_c(a) \leqq DN^a.$$

In other words, let us find $c$ such that $\varphi_c(x) \leqq .999$ for all $x \geqq a$, and $a$ as small as possible.

For $c = 3.788$ we find that $a = 10.09425$. To complete the induction proof, set $D = 2$ and observe that $L(n) \leqq Dn^a$ for $n \leqq 2$, while for $n \geqq 3$ we have

$$L(n) \leqq 1 + Dn^a \varphi_c(a) \leqq Dn^a,$$

since $n^a \geqq 3^{10} \geqq 500$.

Thus we have shown the following theorem.

THEOREM 6.1. $L(n) \leqq 2n^{10.1}$.

Using (7), we easily obtain Corollary 6.2.

COROLLARY 6.2. $R(x) \leqq 2x^{11.1}$.

## 7. Open problems.

We have shown that our algorithm for computing $b_w(n)$ has a worst-case running time of $O(|w|^{11.1})$. On the other hand, the worst case we can explicitly construct takes $\Omega(|w|^3)$.

We saw that for a substantial fraction of all $w$, the algorithm actually runs in linear time. Thus, a natural question to ask is: *What is the expected running time of our algorithm?*

Our results have also shown that the degree $d_w$ of the numerator and denominator of the rational function $b_w(n)$ satisfies $d_w \leqq 2|w|^{10.1}$. However, the worst case we can explicitly construct has $d_w = \lfloor 3|w|/2 \rfloor - 2$.

**Appendix.** *Values of* $b_w(n)$.

| $w$ | $b_w(n)$ |
|---|---|
| 0 | $\frac{2n}{2n+1}$ |
| 1 | $\frac{2n+1}{2n+2}$ |
| 00 | $\frac{2n}{2n+1} \cdot \frac{4n+3}{4n+2}$ |
| 01 | $\frac{4n+1}{4n+2}$ |
| 10 | $\frac{4n+2}{4n+3}$ |
| 11 | $\frac{2n+1}{2n+2} \cdot \frac{4n+2}{4n+1}$ |
| 000 | $\frac{2n}{2n+1} \cdot \frac{4n+3}{4n+2} \cdot \frac{8n+5}{8n+4}$ |
| 001 | $\frac{8n+1}{8n+2}$ |
| 010 | $\frac{4n+2}{4n+3} \cdot \frac{8n+7}{8n+6}$ |
| 011 | $\frac{8n+3}{8n+4}$ |
| 100 | $\frac{8n+4}{8n+5}$ |
| 101 | $\frac{4n+1}{4n+2} \cdot \frac{8n+2}{8n+1}$ |
| 110 | $\frac{8n+6}{8n+7}$ |
| 111 | $\frac{2n+1}{2n+2} \cdot \frac{4n+2}{4n+1} \cdot \frac{8n+4}{8n+3}$ |

**Acknowledgment.** We thank Andrew Odlyzko for helping us locate the reference [Nie].

## REFERENCES

[ACMS] J.-P. ALLOUCHE, H. COHEN, M. MENDÈS FRANCE, AND J. O. SHALLIT, *De nouveaux curieux produits infinis*, Acta Arith., 49 (1987), pp. 141–153.

[AS] J.-P. ALLOUCHE AND J. O. SHALLIT, *Infinite products associated with counting blocks in binary strings*, J. London Math. Soc., to appear.

[Nie] P. TOLSTRUP NIELSEN, *A note on bifix-free sequences*, IEEE Trans. Inform. Theory, 19 (1973), pp. 704–706.

[Rob] D. ROBBINS, *Solution to problem* E2692, Amer. Math. Monthly, 86 (1979), pp. 394–395.

[Sha] J. O. SHALLIT, *On infinite products associated with sums of digits*, J. Number Theory, 21 (1985), pp. 128–134.

[Woo] D. R. WOODS, *Elementary problem proposal* E2692, Amer. Math. Monthly, 85 (1978), p. 48.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 16-27, February 1989

# INTERSECTION OF TWO MATROIDS: (CONDENSED) BORDER GRAPHS AND RANKING*

PAOLO M. CAMERINI† AND HORST W. HAMACHER‡

**Abstract.** Given two matroids $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$, three algorithms for finding $K$ best intersections $I_1, I_2, \cdots, I_K$ are presented. The first version is a straightforward application of a general procedure due to Murty and Lawler. The complexity for finding $I_2, \cdots, I_k$ is $O(Km^2R(R + c(m) + \log m))$ where $m$ is the number of elements in $E$, $R = \min\{r_1(E), r_2(E)\}$, and $c(m)$ is the complexity of an independence oracle. By using maximum weighted border paths to compute second best intersections of modified matroids, this bound is reduced to $O(K(m^3 + mRc(m)))$. Finally, a condensed version of the border graph is proposed to further improve the bound to $O(KmRc(m))$. The latter idea can also be used to find the optimal intersection $I_1$ in $O(mR^2c(m))$ time, which is competitive with recent matroid intersection algorithms by Frank [*J. Algorithms*, 2 (1981), pp. 328-336] and Brezovec, Cornuejols, and Glover [*Math. Programming*, 36 (1986), pp. 39-53].

**Key words.** matroid intersection, ranking, efficient algorithms

**AMS(MOS) subject classifications.** 05B35, 90C10

## 1. Introduction.

Let $M_1 = (E, \mathcal{I}_1)$, $M_2 = (E, \mathcal{I}_2)$ be two *matroids* with the same *ground set* $E = \{e_1, \cdots, e_m\}$ of *elements* and with nonempty families $\mathcal{I}_i \subseteq 2^E$ of *independent sets* satisfying

$$I \in \mathcal{I}_i, J \subseteq I \rightarrow J \in \mathcal{I}_i, \tag{1.1}$$

$$I, J \in \mathcal{I}_i, |I| = |J| + 1 \rightarrow \exists e \in I - J : J + e \in \mathcal{I}_i \tag{1.2}$$

for $i = 1, 2$. (As usual, in matroid theory we use the denotation $J + e$ as an abbreviation for $J \cup \{e\}$.) For these and other standard matroid notation and properties we refer to Lawler [19]. For $i = 1, 2$ let $r_i(A)$ and $sp_i(A)$ denote the *rank* and the *span*, respectively, of any subset $A$ of $E$ in matroid $M_i$ and denote $R = \min\{r_1(E), r_2(E)\}$. In the following we assume for computational purposes that the two matroids are given in *concise form*; i.e., the input length needed for their specifications grows no more than polynomially with the size $m$ of the ground set. Moreover we assume each matroid $M_i$, $i = 1, 2$, is specified by means of an *independence oracle*; i.e., we assume to know a procedure that decides in time bounded by the polynomial $c_i(m)$, whether or not a given set $A \subseteq E$ is independent in $M_i$. We denote $c(m) = \max\{c_1(m), c_2(m)\}$. To simplify the exposition we also assume without loss of generality that $E \neq \phi$ and both matroids are *normal* (i.e., each singleton $\{e\}$ is independent).

For each $e_j \in E$ let $w_j = w(e_j)$ denote a known integer number – the *weight* of $e_j$. Let

$$w(A) = \sum_{e_j \in A} w_j \tag{1.3}$$

* Received by the editors June 30, 1986; accepted for publication (in revised form) May 6, 1988.

† Dipartimento di Elettronica, Centro di Studio per le Telecomunicazioni Spaziali del CNR, Politecnico di Milano, Milano, Italy. This research was supported in part by the North Atlantic Treaty Organization under grant RG 85/0240 and research contract MPI on "Analysis and Design of Algorithms" of the Italian Ministry of Education.

‡ Fachbereich Mathematik, Universität Kaiserslautern, Kaiserslautern, West Germany, and Center for Optimization and Combinatorics, University of Florida, Gainesville, Florida 32611. This research was supported in part by the North Atlantic Treaty Organization under grant RG 85/0240 and by National Science Foundation grant ECS-8412926.

be the weight of any subset $A \subseteq E$ with the understanding that $w(\phi) = 0$. We use the terminology *best*, and *better* to indicate that the weight of a set is maximum, and not less, respectively, compared with other sets.

Let $\mathscr{I} = \mathscr{I}_1 \cap \mathscr{I}_2$ be the family of all *intersections* of the two matroids $M_1$ and $M_2$. A sequence $I_1, \cdots, I_K$ of $K \geqq 1$ distinct members of $\mathscr{I}$ is called a sequence of $K$ *best intersections* if $w(I_k) \geqq w(I_{k+1})$, $k = 1, \cdots, K-1$, (when $K > 1$) and $I_K$ is better than any other member of $\mathscr{I}$. Consequently, $I_k$ is called a $k$th *best intersection*, $k = 1, \cdots, K$, of $M_1$ and $M_2$. The *matroid intersection ranking problem* is the following problem. Find $K$ best intersections for given matroids $M_1$, $M_2$, weights $w_j$, $j = 1, \cdots, m$, and integer $K \geqq 1$. For $K = 1$ this problem includes the best intersection problem that has been solved in [7], [16], and [18] by $O(m^2R^2 + mR^2c(m))$ algorithms (see, for instance [19, Chap. 8]), and with an $O(mR^2 + mRc(m) + mR \log m)$ algorithm in [1].

In the next section we show that if $K \geqq 2$, then $I_2, \cdots, I_K$ can be found in $O(K(R^2m^2 + Rm^2c(m) + Rm^2 \log m))$ time by a straightforward application of the Lawler-Murty approach [17], [21], using the matroid intersection algorithm of Brozovec et al. [1]. This bound is improved in § 3, where we prove a key property of the border graph, allowing us to obtain $I_2$ from a best intersection $I_1$ in $O(m^3 + mRc(m))$ time. Using this fact and taking advantage of a binary tree version of the Lawler-Murty procedure developed in [10] and [11], we show that $I_2, \cdots, I_K$ can be found in $O(K(m^3 + mRc(m)))$ time. This complexity bound is further reduced in § 4 to $O(KmRc(m))$ by computing all pairs shortest paths in a condensed version of the border graph. This technique can also be utilized in order to compute $I_1$ in $O(mR^2c(m))$ time, thus improving the primal-dual implementation of the algorithm given in [19, Chap. 8].

**2. The Lawler-Murty procedure for $K$ best intersections.** In [20] Murty developed a procedure for finding $K$ best assignments in a given weighted bipartite graph. This procedure has been generalized by Lawler [17] to arbitrary combinatorial optimization problems. For applications of this procedure to a specific problem see, e.g., [3], [4], [14], and [20]. In this section we apply the Lawler-Murty procedure to obtain a straightforward algorithm for finding $K$ best intersections [22].

Let $X \subseteq E$ be an arbitrary subset of $E$ and let $Y \in \mathscr{I}$ be any intersection such that $X \cap Y = \phi$. Denote

$$\mathscr{I}(X, Y) = \{I \in \mathscr{I} \mid I \cap X = \phi, Y \subseteq I\}. \tag{2.1}$$

Since $Y \in \mathscr{I}$, the contraction of $Y$ and the deletion of $X \cup sp_1(Y) \cup sp_2(Y)$ in $\mathscr{I}_i$, $i = 1, 2$, yield the following normal matroids for $i = 1, 2$:

$$M_i(X, Y) := (M_i \text{ ctr } Y)\text{del}(X \cup sp_1(Y) \cup sp_2(Y) - Y) = (E(X, Y), \mathscr{I}_i(X, Y)) \tag{2.2}$$

where

$$E(X, Y) = E - (X \cup sp_1(Y) \cup sp_2(Y)),$$

$$\mathscr{I}_i(X, Y) = \{I' \subseteq E(X, Y) \mid I' \cup Y \in \mathscr{I}_i\}.$$

Since $X \cap Y = \phi$, it follows that

$$\mathscr{I}(X, Y) = \{I' \cup Y \mid I' \in (\mathscr{I}_1(X, Y) \cap \mathscr{I}_2(X, Y))\}. \tag{2.3}$$

Moreover, $\mathscr{I}_l(X, Y)$, the $l$th best member of $\mathscr{I}(X, Y)$ $(l \geqq 1)$, has the form

$$I_l(X, Y) = I'_l \cup Y \tag{2.4}$$

where $I'_l$ is an $l$th best intersection of $M_1(X, Y)$ and $M_2(X, Y)$, since the weight of each member of $\mathscr{I}(X, Y)$ and of its (by (2.3)) corresponding intersection of $M_1(X, Y)$ and $M_2(X, Y)$ differ only by the constant term $w(Y)$. Therefore we obtain for $l \geqq 1$,

$I_l(X, Y)$ is an $l$th best intersection in $\mathscr{I}(X, Y)$ if and only if

(2.5) $I_l(X, Y) = I'_l \cup Y$ and $I'_l$ is an $l$th best intersection of

$M_1(X, Y)$ and $M_2(X, Y)$.

$I_1(X, Y)$ can therefore be found in $O(mR(R + c(m) + \log m))$ time by using the matroid intersection algorithm of [1], since deletion and contraction operations do not affect the complexity of the independence oracle.

The Lawler–Murty procedure, applied to matroid intersection problems, can now be described as follows. Suppose we have found that the $k$th best intersection $I_k$ is the best intersection in $\mathscr{I}(X_k, Y_k)$ for $X_k \subset E$, and $Y_k \in \mathscr{I}$. Then $\mathscr{I}(X_k, Y_k)$ is further partitioned by iteratively excluding elements $e \in (I_k - Y_k)$ from intersections and forcing elements $e \in E - (I_k \cup X_k \cup sp_1(Y_k) \cup sp_2(Y_k))$ into intersections. For each pair $(X, Y)$ the best intersection $\mathscr{I}_1(X, Y)$ is computed and added to the candidate list $\mathscr{L}$. The next best intersection $I_{k+1}$ is found by taking an intersection in $\mathscr{L}$ with the largest weight. The validity of the algorithm subsequently detailed follows from [17].

ALGORITHM FOR RANKING MATROID INTERSECTIONS.

```
    input: normal matroids $M_1 = (E,\mathscr{I}_1)$, $M_2 = (E,\mathscr{I}_2)$ in concise form,
           integer weight for each $e_j \in E$,
           positive integer $K$;
    output: $K$ best intersections $I_1, \cdots, I_K$ in $\mathscr{I} = \mathscr{I}_1 \cap \mathscr{I}_2$;
    begin
 1. find a best intersection $I_1$ in $\mathscr{I}$;
 2. $k \leftarrow 1$; $X_1 \leftarrow Y_1 \leftarrow \varnothing$; $\mathscr{L} \leftarrow \varnothing$;
 3. while $k < K$ do
      begin
 4.     $U \leftarrow I_k - Y_k$; $V \leftarrow E - (I_k \cup X_k \cup sp_1(Y_k) \cup sp_2(Y_k))$;
 5.     declare unvisited all elements of $U$ and $V$;
 6.     for each $e \in U$ do
          begin
 7.         $X \leftarrow X_k + e$; $Y \leftarrow Y_k \cup \{f \in U \mid f$ has been visited$\}$;
 8.         visit $e$; find $I = I_1(X,Y)$;
 9.         add the triple $(I,X,Y)$ to $\mathscr{L}$
          end;
10.     for each $f \in V$ do
          begin
11.         $Y \leftarrow Y_k + f$; $X \leftarrow X_k \cup \{e \in V \mid e$ has been visited$\}$;
12.         visit $f$; find $I = I_1(X,Y)$;
13.         add the triple $(I,X,Y)$ to $\mathscr{L}$
          end;
14.     if $L = \varnothing$ then stop [$I$ contains only $k < K$ intersections] else
          begin
15.         extract from $\mathscr{L}$ a triple $(I,X,Y)$ such that $I$ has max weight;
16.         $k \leftarrow k+1$; $I_k \leftarrow I$; $X_k \leftarrow X$; $Y_k \leftarrow Y$
        end
      end
    end
```

The LM Algorithm uses $(K-1)$ iterations in which the computation of at most $m$ intersections in lines 8 and 12 is the most time-consuming part. Hence the LM Algorithm for computing $I_2, \cdots, I_K$ has a complexity of $O(K(m^2R^2 + m^2Rc(m) + m^2R \log m))$ if the algorithm of [1] is used to find the maximum weight intersections of Step 12.

**3. Using second best intersections for an improved algorithm.** If we analyze the algorithm of the previous section, we realize that for each value of $k$, lines 8 and 12 of Algorithm LM generate more intersections than we need. Out of $O(m)$ many solutions it is sufficient to consider the best one, which is $I_2(X_k, Y_k)$, and add it to the list. If $I_2(X_k, Y_k)$ can be found by a procedure with worst-case bound better than $O(m^2R(R + c(m) + \log m))$, then this modified procedure will be superior to the algorithm presented in the previous section. This idea has been used successfully in [5], [6], [10], [12], [13], and [11] to develop efficient algorithms for finding $K$ best bases of matroids, cuts in networks, perfect matchings, spanning trees, and arborescences in graphs.

In this section we will show that $I_2(X_k, Y_k)$ can be obtained by finding a maximum weighted border path in a suitable border graph. Using (2.5) with $l = 2$, we can without loss of generality restrict the discussion to the case where $X_k = Y_k = \phi$, i.e., $k = 1$.

Recall that the *border graph* BG $(I)$ of an intersection $I \in \mathcal{I}$ is a directed bipartite graph with arcs connecting nodes representing $E - I$ and $I$ as follows. Let $e_i \in I$, $e_j \in E - I$. Then

$$(3.1) \qquad (e_i, e_j) \in \mathrm{BG}(I) \quad \text{if } (I + e_j) \notin \mathcal{I}_1 \text{ but } ((I + e_j) - e_i) \in \mathcal{I}_1,$$

$$(3.2) \qquad (e_j, e_i) \in \mathrm{BG}(I) \quad \text{if } (I + e_j) \notin \mathcal{I}_2 \text{ but } ((I + e_j) - e_i) + \mathcal{I}_2.$$

The indegree and outdegree of nodes in BG $(I)$ is the *indegree and outdegree of element* $e \in E$ (*with respect to* $I$). Elements $e \in (E - I)$ with indegree (outdegree) 0 are called *sources* (*sinks*).

A *border path* is one of the following five types of directed paths in BG $(I)$ from a node $e_i$ to a (not necessarily different) node $e_j$:

(1) A *source-sink path*, i.e., $e_i$ is a source, and $e_j$ is a sink;
(2) A *source-I path*, i.e., $e_i$ is a source, and $e_j \in I$;
(3) An *I-sink path*, i.e., $e_i \in I$, and $e_j$ is a sink;
(4) An (*open*) *I-I path*, i.e., $e_i \in I$, $e_j \in I$, and either $e_i \neq e_j$ or $e_i = e_j$ is the unique node of the path;
(5) A *cycle*, i.e., $e_i = e_j \in I$.

Note that in (1) and (4), $e_i = e_j$ is possible; i.e., the paths may contain only the single node $e_i = e_j$, whereas for cycles we assume that they contain at least two different nodes. A *shortcut* of a border path $P = (e_{i_1}, \cdots, e_{i_a}, \cdots, e_{i_b}, \cdots, e_{i_q})$ is any arc $(e_{i_a}, e_{i_b})$ in BG $(I)$ with $a < b$.

The *incremental weight* $\Delta(P)$ of a border path $P$ is defined by

$$(3.3) \qquad \Delta(P) = \sum_{e_j \in (P - I_1)} w_j - \sum_{e_j \in (P \cap I_1)} w_j.$$

A shortcut $(e_{i_a}, e_{i_b})$ is a *worsening*, *improving*, and *zero shortcut* for border path $P$ if $\Delta(P') < \Delta(P)$, $\Delta(P') > \Delta(P)$, and $\Delta(P') = \Delta(P)$, respectively, for the shortcutted border path $P' = (e_{i_1}, \cdots, e_{i_a}, e_{i_b}, \cdots, e_{i_q})$.

Finally, we call a border path *primitive* if it has no shortcuts or if all its shortcuts are worsening. With this definition the following lemma has been shown in [15].

LEMMA 3.1. *If $P$ is a primitive border path in* BG $(I_1)$, *then*

$$I_1 \oplus P = (I_1 \cup P) - (I_1 \cap P)$$

*is an intersection. The weight of $I_1 \oplus P$ can be computed by using* (3.3).

$$w(I_1 \oplus P) = w(I_1) + \Delta(P). \tag{3.4}$$

The next lemma establishes that we can restrict ourselves to primitive border paths if we are looking for a border path with maximum incremental weight $\Delta(P)$.

LEMMA 3.2. *The maximum incremental weight of all border paths in* BG $(I_1)$ *is attained in a primitive border path $P_1$ with $\Delta(P_1) \leqq 0$.*

*Proof.* Let $P_1$ such that $\Delta(P_1) = \max\{\Delta(P) \mid P \text{ primitive border path}\}$, and let $S$ be an arbitrary border path. If $S$ is primitive, then $\Delta(P_1) \geqq \Delta(S)$. Otherwise, $S$ admits nonworsening shortcuts. After a finite number of the shortcuts, $S$ is therefore transformed to a primitive border path $S'$ such that $\Delta(S) \leqq \Delta(S')$ and $\Delta(P_1) \geqq \Delta(S)$ follows from the first case.

To show $\Delta(P_1) \leqq 0$, we observe that the assumption $\Delta(P_1) > 0$ together with Lemma 3.1 and (3.4) contradict the fact that $I_1$ is the maximum weighted intersection.

We are now able to prove the main result of this section.

THEOREM 3.1. *If $P_1$ is a primitive border path in* BG $(I_1)$ *with maximum incremental weight $\Delta(P_1)$, then $I_1 \oplus P_1$ is a second best intersection.*

*Proof.* By Lemma 3.1, $I_1 \oplus P_1$ is an intersection and it remains to show that $w(I_1 \oplus P_1) \geqq w(J)$ for all intersections $J \neq I_1$. For that purpose let $J \neq I_1$ be any intersection. From [15] (see also [19, Chap. 8]), we know that the nodes in $I_1 \oplus J$ can be covered by border paths $S_1, \cdots, S_l$. (Note that some of these paths may be isolated nodes in BG $(I_1)$.) Therefore

$$\begin{aligned} w(J) &= (w(I_1 \cap J) + w(I_1 - J)) + (w(J - I_1) - w(I_1 - J)) \\ &= w(I_1) + \sum_{j=1}^{l} \Delta(S_j) \\ &\leqq w(I_1) + l \cdot \Delta(P_1) \\ &\leqq w(I_1) + \Delta(P_1) \\ &= w(I_1 \oplus P_1), \end{aligned}$$

where the inequalities follow from Lemma 3.2. □

Any cycle $C$ in BG $(I_1)$ is a border path. Therefore Lemma 3.2 yields

$$\Delta(C) \leqq \Delta(P_1) \leqq 0; \tag{3.5}$$

i.e., BG $(I_1)$ contains no positive cycle. Hence the Floyd–Warshall Algorithm [8], [24] can be used to find a maximum weighted border path $P$ in BG $(I_1)$. (Note that the weights are on the nodes instead of the arcs, and minor modifications of the Floyd–Warshall Algorithm are necessary.) The complexity of this procedure is $O(m^3)$. $P$ does not admit improving shortcuts. But we may have to remove shortcuts to transform $P$ into a maximum weighted, primitive border path $P_1$. Since these shortcuts are found in $O(m^2)$ time, the overall complexity of finding $P_1$ is $O(m^3)$, and the complexity of finding $I_2, \cdots, I_K$ is $O(K(m^3 + mRc(m)))$, where $O(mRc(m))$ is the time for constructing BG $(I_1)$.

An alternative way to find $P_1$ is based on the following observation. A sufficient condition for a maximum weighted border path $P$ to be primitive is that the number of nodes is minimum. Therefore we find a minimum path with respect to vector-valued node weights $(-w_j, 1)$ and use the lexicographic version of the Floyd–Warshall Algorithm [2], [25] to find $P_1$ without having to look for shortcuts. Since the complexity of this procedure is also $O(m^3)$, the bound for finding $I_2, \cdots, I_K$ is unchanged.

We formalize the algorithm developed in this section as follows.

Binary Algorithm for Ranking Matroid Intersections.

```
    input:  normal matroids $M_1 = (E, \mathscr{I}_1)$, $M_2 = (E, \mathscr{I}_2)$ in concise form,
            integer weight $w_j$ for each $e_j \in E$,
            integer $K \geqq 2$;
    output: $K$ best intersections $I_1, \cdots, I_K$ in $\mathscr{I} = \mathscr{I}_1 \cap \mathscr{I}_2$
    begin
 1. find a best intersection $I_1$ in $\mathscr{I}$;
 2. find in BG $(I_1)$ a primitive border path $P_1$ of maximum incremental weight;
 3. $I_2 \leftarrow I_1 \oplus P_1$; $p \leftarrow 1$; $X_1 \leftarrow Y_1 \leftarrow \varnothing$; $k \leftarrow 2$; $\mathscr{L} \leftarrow \varnothing$;
 4. while $k < K$ do
      begin
 5.     choose any $e \in I_p \oplus I_k$
 6.     if $e \in I_p - I_k$ then begin $Y_k \leftarrow Y_p$; $X_k \leftarrow X_p + e$; $Y_p \leftarrow Y_p + e$ end;
 7.     else $[e \in I_k - I_p]$ begin $X_k \leftarrow X_p$; $Y_k \leftarrow Y_p + e$; $X_p \leftarrow X_p + e$ end;
 8.     for each $j \in \{p,k\}$ do if $E(X_j, Y_j) \neq \phi$ then
          begin
 9.         construct BG $(I_j - Y_j)$ with respect to matroids $M_i(X_j,Y_j)$, $i = 1,2$;
10.         find in BG $(I_j - Y_j)$ a primitive border path $P_j$ of maximum
            incremental weight [e.g., by using the lexicographic Floyd-
            Warshall Algorithm with vector valued weight $(-w_h, 1)$
            associated with each node $e_h$];
11.         add the quadruple $(I_j \oplus P_j; X_j, Y_j, j)$ to $\mathscr{L}$
            [$I_j$ is $I_1$ $(X_j, Y_j)$ and $I_j \oplus P_j$ is $I_2$ $(X_j, Y_j)$]
          end;
12.     if $\mathscr{L} = \phi$ then stop [$\mathscr{L}$ contains only $k < K$ intersections] else
          begin
13.         extract from $\mathscr{I}$ a quadruple $(I,X,Y,p)$ such that $I$ has maximum weight;
14.         $k \leftarrow k+1$; $I_k \leftarrow I$; $X_k \leftarrow X$; $Y_k \leftarrow Y$
          end
      end
    end
```

**4. Condensed border graphs and their applications.** For the following we assume $I$ is an intersection that does not admit positive cycles in its border graph, i.e.;

(4.1) $\Delta(C) = w(C-I) - w(C \cap I) \leqq 0$ for all cycles $C$ in BG $(I)$.

The *condensed border graph* CB $(I)$ of $I$ is in general a nonbipartite, directed graph with node set $I \cup \{q, s, t\}$ and arc set $A$ connecting the nodes $e_i$, $e_j$, $q$, $s$, and $t$ as follows:

(4.2) $(e_i, e_j) \in A \Leftrightarrow \exists e \in (E-I): (e_i, e), (e, e_j) \in \mathrm{BG}(I)$

(note that $(e_i, e_i) \in A$ is possible, i.e., CB$(I)$ may contain loops),

(4.3) $(s, e_i) \in A \Leftrightarrow \exists$ source $e \in (E-I): (e, e_i) \in \mathrm{BG}(I)$,

(4.4) $(q, e_i) \in A \quad \forall e_i \in I$, (4.5) $(e_i, t) \in A \Leftrightarrow \exists$ sink $e \in (E-I)\colon (e_i, e) \in \mathrm{BG}(I)$,

(4.6) $(s, t) \in A \Leftrightarrow \exists e \in E - S$: $e$ is both a source and a sink.

The main goal of this section is to establish that border paths used in computing $I_1, I_2, \cdots, I_K$ are in one-to-one correspondence with paths in some CB $(I)$, and to replace the search for a primitive, maximum weighted border path in BG $(I)$ by a simpler procedure to find a maximum weighted path in CB $(I)$. For this purpose we define weights for the arcs $a \in A$.

For arcs $(e_i, e_j) \in A$ defined by (4.2) we call $e \in (E - I)$ an *intermediate node* of $e_i$, $e_j \in I$ if $(e_i, e), (e, e_j) \in \mathrm{BG}(I)$. For each $(e_i, e_j)$ the set of intermediate nodes is nonempty by (4.2) and we denote with $\phi_{ij} = \phi(e_i, e_j)$ an intermediate node with maximum weight $w(\phi_{ij})$. The weight of $(e_i, e_j)$ in CB $(I)$ is

(4.7) $$\delta(e_i, e_j) = w(\phi_{ij}) - w(e_j).$$

If $a = (s, e_i) \in A$, there exists by (4.3) at least one source $e \in (E - I)$, such that $(e, e_i) \in \mathrm{BG}(I)$. We call each of these sources a *source for* $e_i$ and denote with $\phi_{si} = \phi(s, e_i)$ a source for $e_i$ with maximum weight. The weight of $(s, e_i)$ is

(4.8) $$\delta(s, e_i) = w(\phi_{si}) - w(e_i).$$

For $a = (q, e_i)$ defined by (4.4), we set

(4.9) $$\delta(q, e_i) = -w(e_i).$$

Any sink $e$ in BG $(I)$ connected with $e_i \in I$ by an arc $(e_i, e)$ is a *sink for* $e_i$ and we denote with $\phi_{it} = \phi(e_i, t)$ a sink for $e$ with maximal weight. Then

(4.10) $$\delta(e_i, t) = w(\phi_{it}).$$

Finally, if $(s, t) \in A$, we denote with $\phi_{st} = \phi(s, t)$ a most weighted element in $E - I$ that is both a source and a sink. Then

(4.11) $$\delta(s, t) = w(\phi_{st}).$$

For a given border graph BG $(I)$, CB $(I)$ and the weights $\delta(a)$ can be computed in $O(mR^2)$ time. The construction of BG $(I)$ itself has a complexity of $O(mRc(m))$. Since in this construction the input of each independence oracle is $(J + e)$ where $J \subseteq I$ and $e \in E - I$, we have $c(m) = \Omega(R)$, and the overall complexity of computing BG $(I)$, CB $(I)$ and the weights $\delta(a)$ is $O(m \cdot R \cdot c(m) + mR^2) = O(mRc(m))$.

Note that we can specify the ambiguous definitions of $\phi_{ij}$, $\phi_{si}$, $\phi_{it}$, and $\phi_{st}$ by introducing any tie-breaking rule. We could, for instance, take the maximum weighted intermediate node (source, sink) with the smallest index.

*Example* 4.1. Let $M_i$ be the graphic matroids of the graphs $G_i$, $i = 1, 2$, as shown in Fig. 1. The weights are $w(e_j) = w_j = 11 - 2[(j+1)/2]$, $j = 1, \cdots, 9$. $I = \{e_1, e_2, e_3\}$ is independent in both matroids, and the border graph BG $(I)$ is shown in Fig. 2. The condensed border graph CB $(I)$ is shown in Fig. 3. To find, for instance, $\phi_{s1}$, we compute $\max\{w_4, w_5\} = 7$ yielding $\phi_{s1} = e_4$, and $\delta(s, e_1) = w_4 - w_1 = 7 - 9 = -2$. Similarly, $\phi_{12} = e_6$ and $\delta(e_1, e_2) = -4$.

In this example we may check that the five different types of border paths $P$ between two nodes $e_i$ and $e_j$ in BG $(I)$ correspond to paths $Q = Q(P)$ in the condensed border graph:

(4.12) from $s$ to $t$ if $P$ is a source-sink path,

(4.13) from $s$ to $e_i$, if $P$ is a source-$I$ path,

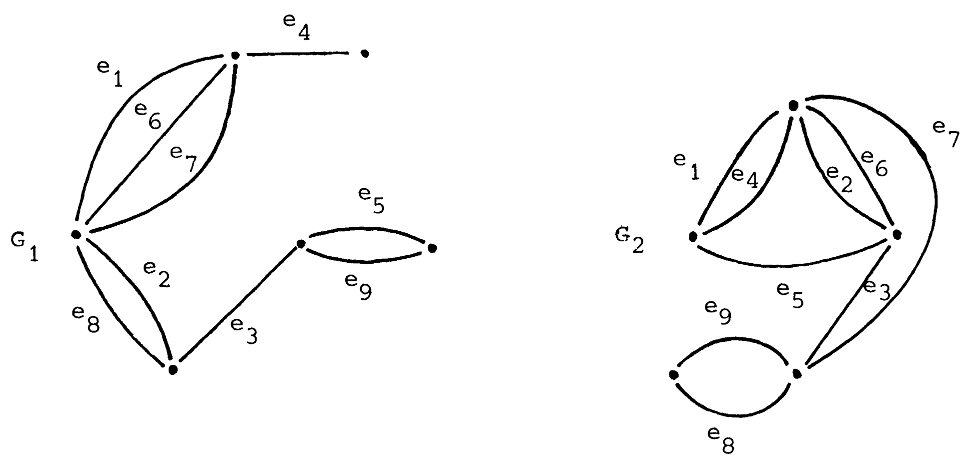


FIG. 1. *Graphs defining graphic matroids.*

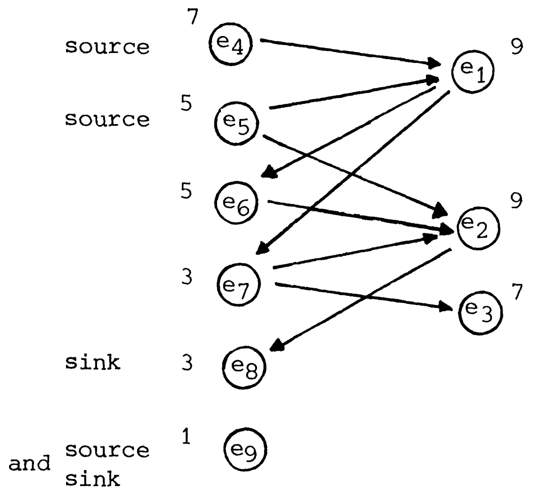


FIG. 2. *Border graph* BG ($I$) *with node weights* $w_j$.

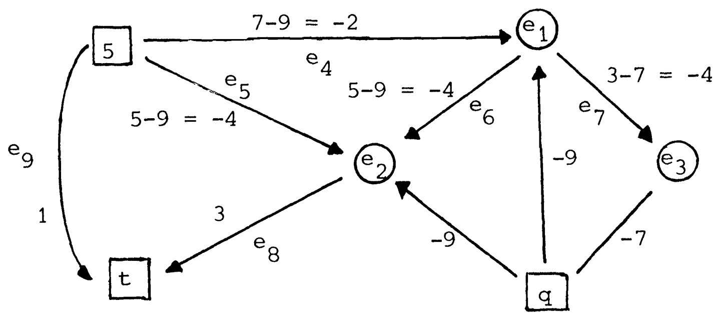


FIG. 3. CB ($I$) *with arc weights* $\delta$ *and with* $\phi_{ij}$ *on the arcs.*

(4.14) from $q$ to $t$ if $P$ is a $I$-sink path,

(4.15) from $q$ to $e_i$ if $P$ is an open $I$-$I$ path.

A fifth type of path $Q(P)$, namely

(4.16) from $e_i$ to $e_i$ if $P$ is a cycle

is not contained in this particular example. The weights of corresponding paths satisfying $\Delta(P) = w(P - I) - w(P \cap I) \leqq \delta(Q(P)) = \sum_{a \in Q(P)} \delta(a)$. Suppose in this example we want to find a source-sink border path with maximum incremental weight, which is $P = (e_9)$. This path corresponds to $Q(P) = (s, t)$ in CB $(I)$, which has the same weight. In the same fashion the path $P = (e_4, e_1, e_6, e_2, e_8)$ with

$$\Delta(P) = w(P - I) - w(P \cap I) = 15 - 18 = -3$$

corresponds to the $(s, t)$ path $Q(P) = (s, e_1, e_2, t)$ with $\delta(Q(P)) = \Delta(P) = -3$.

In general, let

$$P = (e_{i_0}, e_{i_1}, e_{i_2}, \cdots, e_{i_{j-1}}, e_{i_j}, e_{i_{j+1}}, \cdots, e_{i_{l-1}}, e_{i_l}, e_{i_{l+1}})$$

be an arbitrary border path in BG $(I)$. We assume in this denotation that $e_{i_j} \in E - I$ and $e_{i_0} = e_{i_{l+1}} \in I$, if $P$ is a cycle. Then

$$Q(P) = (f_1, f_2, \cdots, e_{i_{j-1}}, e_{i_{j+1}}, \cdots, d_1, d_2)$$

is a path in CB $(I)$ where $f_1, f_2$ and $d_1$, $d_2$ are defined according to Table 1 depending on the type of path we are considering. Therefore each border path $P$ corresponds to a path $Q(P)$ in CB $(I)$ and from the definition of $\delta$ we get

(4.17) $$\Delta P \leqq \delta(Q(P)).$$

Also note that Table 1 defines a *type correspondence* between paths $P$ in BG $(I)$ and $Q(P)$ in CB $(I)$. For instance a Type (3) path in BG $(I)$ is an $I$-sink path whereas a Type (3) path in CB $(I)$ is a $q$-$t$ path.

Since we are trying to replace the search for primitive, maximum incremental weighted paths $P$ in BG $(I)$ by a path computation in CB $(I)$, we must study the converse situation, i.e., how to construct a path $P = P(Q)$ in BG $(I)$ from a given path $Q$ in CB $(I)$. To do that, we consider the following two cases.

*Case* 1.

(4.18) $$a \neq b \Rightarrow \phi(a) \neq \phi(b) \quad \forall a, b \in Q.$$

In this case $Q = (f_1, f_2, \cdots, e_{i_{j-1}}, e_{i_{j+1}}, \cdots, d_1, d_2)$ implies that a path $P = P(Q)$ in BG $(I)$ can be found by using Table 1 and adding the nodes $\phi(e_{i_{j-1}}, e_{i_{j+1}})$ between

TABLE 1

| $e_{i_0}$ <br> $e_{i_{l+1}}$ | source <br> sink | source <br> $\in I$ | $\in I$ <br> sink | $\in I$ <br> $\in I$ <br> (open $I$-$I$ path) | $\in I$ <br> $\in I$, $e_{i_{l+1}} = e_{i_0}$ <br> (cycle) |
|---|---|---|---|---|---|
| $f_1$ | $s$ | $s$ | $q$ | $q$ | $e_{i_0}$ |
| $f_2$ | $e_{i_1}$ | $e_{i_1}$ | $e_{i_0}$ | $e_{i_0}$ | $e_{i_2}$ |
| $d_1$ | $e_{i_{l+1}}$ | $e_{i_{l-1}}$ | $e_{i_{l+1}}$ | $e_{i_{l-1}}$ | $e_{i_{l-1}}$ |
| $d_2$ | $t$ | $e_{i_{l+1}}$ | $t$ | $e_{i_{l+1}}$ | $e_{i_{l+1}}$ |
| | Type (1) | Type (2) | Type (3) | Type (4) | Type (5) |

$e_{i_{j-1}}$ and $e_{i_{j+1}}$. Moreover (4.17) holds in this case with equality by the definition of the weights $\delta(a)$ in (4.7)–(4.11). Now assume $Q$ is a maximum weighted path in CB $(I)$. Since we have already shown that each path $P$ in BG $(I)$ yields a path $Q(P)$ in CB $(I)$, $P(Q)$ is therefore a maximum weighted border path. After removing from $P(Q)$ all zero shortcuts, we obtain a primitive path $P'$ with maximum incremental weight. (Note that $P(Q)$ does not admit improving shortcuts, because $P(Q)$ is a maximum weighted border path.)

*Case* 2.

(4.19) $$\exists a,b \in Q\colon\quad a \neq b \quad\text{and}\quad \phi(a)=\phi(b).$$

(See Fig. 4.) If $a = (e_i, e_j)$ and $b = (e_h, e_k)$, the definition of the arc set $A$ implies that $c = (e_i, e_k)$, $d = (e_h, e_j) \in A$. We then define the subpaths $Q_1$, $Q_2$, and $Q_3$ of $Q$ as indicated in Fig. 4, to get $Q' = (Q_1, c, Q_3)$, a path of the same type as $Q$, and $(Q_2, d)$, a cycle. If $Q'$ satisfies the condition of Case 1 we stop the procedure. Otherwise we iterate with $Q = Q'$. After at most $|Q| = O(R)$ iterations we end with a path $Q'$ which is of the same type as $Q$ and with, say, $l$ cycles $C_1, \cdots, C_l$ in CB $(I)$ such that

(4.20) $$\delta(Q) \leqq \delta(Q') + \sum_{i=1}^{l} \delta(C_i).$$

For each of the cycles $C_i$ we can also assume that condition (4.18) is satisfied. (If not, we apply the previous procedure to $Q = C_i$, the only consequence being $l$ gets larger.) Hence each cycle $C_i$ in CB $(I)$ corresponds to a cycle $P(C_i)$ in BG $(I)$ and, since (4.17) holds with equality, by assumption (4.1) we get

(4.21) $$\delta(C_i) = \Delta(P(C_i)) \leqq 0, \qquad i = 1, \cdots, l.$$

Now assuming $Q$ is a maximum weighted path in CB $(I)$, it follows that $P(Q')$ is a maximum weight border path in BG $(I)$, because

(4.22) $$\Delta(P(Q')) = \delta(Q') \geqq \delta(Q) \geqq \delta(Q(P)) \geqq \Delta P$$

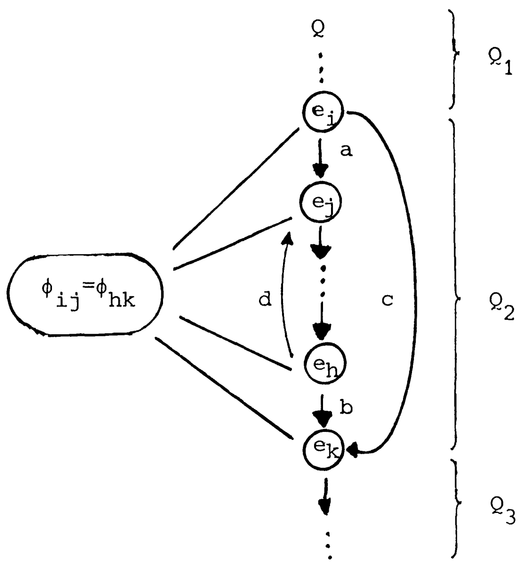


FIG. 4

for all border paths $P$ in BG ($I$). Hence (taking $P = P(Q')$) all inequalities in (4.21) and (4.22) hold with equality and

$$\delta(Q') = \delta(Q). \tag{4.23}$$

If $P(Q')$ has zero shortcuts, then shortcut $P(Q')$ is used to obtain a primitive, maximum weighted border path $P$ in BG ($I$).

From the previous considerations we obtain the following results.

THEOREM 4.1. *If $Q_1$ is a maximum weighted path in* CB ($I$), *then $P(Q_1')$ is a maximum weighted border path in* BG ($I$).

THEOREM 4.2. *For all cycles $C$ in* CB ($I$), $\delta(C) \leqq 0$.

*Proof*. The proof follows from $Q = C$ and (4.21) if we continue decomposing $Q$ in (4.20) until $Q' = \varnothing$.

By Theorem 4.2 we can use the Floyd–Warshall algorithm to compute $Q_1$. Since the decomposition of $Q_1$ according to (4.20) and the computation of $P(Q')$ are dominated by the complexity of this algorithm, $P(Q_1)$ is computed in $O(R^3)$ time. Since $P(Q_1)$ may have zero shortcuts and their removal would add $O(m^2)$ to the complexity of finding $P_1$, the resulting bound $O(R^3 + m^2)$ can be improved by using lexicographic optimization, as the following theorem shows.

THEOREM 4.3. *If $Q_1$ is a lexicographically minimum path in* CG ($I$) *with respect to vector-valued weights* $(-\delta(a), 1)$, *then $Q_1$ satisfies* (4.18) *and $P_1 = P(Q_1)$ is a primitive, maximum weighted border path in* BG ($I$).

*Proof*. The definition of the lexicographical minimum and of the vector-valued weights ensure that $Q_1$ is a maximum weighted path in CB ($I$) with a minimum number of arcs. Hence (4.20) and (4.23) yield that $Q_1$ satisfies (4.18). If $P(Q_1)$ would admit a zero shortcut, then $Q(P')$ where $P'$ is a shortcutted path of $P(Q_1)$ would have the same weight as $Q_1$, but a smaller number of arcs. This contradicts the lexicographic optimality of $Q_1$. □

Using Theorem 4.3 we can do the computation of $P_1$ in $O(R^3)$ time if BG ($I$) and CB ($I$) are given. Thus we obtain the following complexity results.

$I_1$ can be computed by starting with $I = \phi$ and finding a maximum weighted borderpath $P_1$ in BG ($I$) until, for the first time, $\Delta(P_1) \leqq 0$. Note that each of these border paths is necessarily a source-sink path. Therefore this procedure iterates at most $R$ times where each iteration includes the computation of BG ($I$), CB ($I$) and $P_1$, which can be done in $O(mRc(m) + R^3) = 0\,(mRc(m))$. Hence $I_1$ can be computed in $O(mR^2c(m))$.

Note that this bound is better than $O(m^2Rc(m))$ of [9]. Compared with the $O(mR^2 + mRc(m) + mR \log m)$ bound of [1], the condensed border graph algorithm is competitive if $Rc(m) = \theta(\log m)$.

Using the Binary Algorithm of § 3 and the condensed border graph concept to compute $P_j$ in line 10, we have that each intersection $I_k$, $k = 2, \cdots, K$, can be computed in $O(mRc(m))$ time. Hence the complexity for finding $I_2, \cdots, I_K$ is $O(KmRc(m))$.

## REFERENCES

[1] C. BREZOVEC, G. CORNUEJOLS, AND F. GLOVER, *Two algorithms for weighted matroid intersection*, Math. Programming, 36 (1986), pp. 39–53.

[2] P. BRUCKER, *Theory of matrix algorithms*, in Mathematical Systems in Economics 13, Hain, Meisenheim am Glan, West Germany, 1974.

[3] P. M. CAMERINI, L. FRATTA, AND F. MAFFIOLI, *The K shortest spanning trees of a graph*, Internal Report 73-10, IEEE-LCE Polytechnico di Milano, Italy, 1974.

[4] ———, *Efficient ranking of spanning trees and arborescences*, Internal Report IEEE-LCE 75-1, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1975.

[5] ———, *The K best spanning arborescences of a network*, Networks, 10 (1980), pp. 91–110.

[6] C. R. CHEGIREDDY AND H. W. HAMACHER, *Algorithms for K-best perfect matchings*, Discrete Appl. Math., 18 (1987), pp. 155–165.

[7] J. EDMONDS, *Submodular functions, matroids, and certain polyhedra*, in Combinatorial Structures and Their Applications, Proc. Calgary International Conference, R. Guy et al., eds., Gordon and Breach, New York, 1970, pp. 69–87.

[8] R. W. FLOYD, *Algorithm* 97, *shortest path communications*, Assoc. Comput. Mach., 5 (1962), p. 345.

[9] A. FRANK, *A weighted matroid intersection algorithm*, J. Algorithms, 2 (1981), pp. 328–336.

[10] H. N. GABOW, *Two algorithms for generating weighted spanning trees in order*, SIAM J. Comput., 6 (1977), pp. 139–150.

[11] H. HAMACHER AND M. QUEYRANNE, *K best solutions to combinatorial optimization problems*, Ann. Oper. Res., 4 (1986), pp. 123–143.

[12] H. HAMACHER, J. C. PICARD, AND M. QUEYRANNE, *Ranking the cuts and cut-sets of a network*, Ann. Discrete Appl. Math., 19 (1984), pp. 183–200.

[13] ———, *On finding the K best cuts in a network*, Oper. Res. Lett., 2 (1984), pp. 303–305.

[14] H. ISHII, *A new method finding the K-th best path in a graph*, J. Oper. Res. Soc. Japan, 21 (1978), pp. 469–475.

[15] S. KROGDAHL, *A combinatorial proof of Lawler's matroid intersection algorithm*, unpublished manuscript, 1975.

[16] E. L. LAWLER, *Optimal matroid intersections*, in Combinatorial Structures and Their Applications, Proc. Calgary International Conference, R. Guy et al., eds., Gordon and Breach, New York, 1970, p. 233.

[17] ———, *A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem*, Management Sci., 18 (1972), pp. 401–405.

[18] ———, *Matroid intersection algorithms*, Math. Programming, 9 (1975), pp. 31–56.

[19] ———, *Combinatorial Optimization: Networks and Matroids*, Holt, Reinhart and Winston, New York, 1976.

[20] K. G. MURTY, *An algorithm for ranking all the assignments in increasing order of cost*, Oper. Res., 16 (1968), pp. 682–687.

[21] ———, *Solving the fixed change problem by ranking the extreme points*, Oper. Res., 16 (1968), pp. 268–279.

[22] ———, private communication, 1985.

[23] J. B. ORLIN AND J. VANDEVATE, *On a "primal" matroid intersection algorithm*, Working Paper 1446-83, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, 1983.

[24] S. WARSHALL, *A theorem on Boolean matroids*, J. Assoc. Comput. Mach. 9 (1962), pp. 11–12.

[25] U. ZIMMERMANN, *Linear and combinatorial optimization in ordered algebraic structures*, Ann. Discrete Math., 10 (1981).

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 28–37, February 1989

# CODES FROM SYMMETRY GROUPS, AND A [32, 17, 8] CODE*

YING CHENG† AND N. J. A. SLOANE‡

**Abstract.** Let $G$ be the automorphism group of the four-dimensional cube, a group of order $2^4 \cdot 4! = 384$. The binary codes associated with the 32-dimensional permutation representation of $G$ on the edges of the cube are investigated. There are about 400 such codes, one of which is a [32, 17, 8] code, having twice as many codewords as the [32, 16, 8] extended quadratic residue code.

**Key words.** codes, error-correcting codes, permutation groups, modular representations

**AMS(MOS) subject classifications.** 94B, 20C

**1. Introduction.** Since random codes are good ([1], [27, p. 558], [29]), one wishes to identify families of codes large enough to have a chance of including some good codes, yet small enough to be manageable. In this paper we describe one such family: the codes obtained from the action of the automorphism group of the $n$-dimensional cube on its $m$-dimensional faces.

In particular, the automorphism group $G$ of the four-dimensional cube, a group of order 384, permutes the 32 edges of that cube. Regarding the edges as a basis, we have a 32-dimensional vector space $V$ over $GF(2)$ on which $G$ acts. The codes we consider are the subspaces of $V$ invariant under $G$. There are about 400 such subspaces, one of which is a [32, 17, 8] binary code.

We find this quite astonishing, since the well-known second-order Reed–Muller and extended quadratic-residue codes of length 32 are [32, 16, 8] codes, and are extremal Type II self-dual codes ([16], [17, p. 194], [24]). It is remarkable that there should be a linear code with the same minimal distance and twice as many codewords. Of course the new code is not self-dual. Its properties are summarized in Theorem 1.

This family of codes can be generalized in several ways. Besides varying the dimensions of the cube and the faces, we could consider other regular polytopes instead of the cube, or more generally other Weyl groups (our group $G$ is the Weyl group of type $B_4$).

Many other codes have been obtained from modular representations of groups in the past. Of course classical cyclic codes arise from the regular representations of cyclic groups, and include a large number of good examples. In the 1960s Berman [4], [5], Camion [11], Delsarte [19], and MacWilliams [25], [26] studied other abelian groups, but (perhaps because of the limitations of the computers available) did not find any especially interesting codes.

In 1975 Lomonaco (see [15]) found a record [45, 13, 16] binary code obtained as an invariant subspace of the regular representation of the group $C_3 \times C_{15}$. In [10], Calderbank and Wales found a [176, 22, 50] code from the Higman–Sims simple group. Brooke [7]–[9] has studied a large number of other simple groups, using Parker's "meat-axe" [28], without, however, finding any new record codes. Representation theory has also been used to construct codes by Liebler [23], Camion [12], Rabizzoni [32], Ward [34], Zlotnik [36], Klemm [21], Charpin [13], [14], Bhattacharya [6], Jensen [20], Wolfmann [35], and Landrock and Manz [22].

However, it seems fair to say that our [32, 17, 8] binary code is the first record code of length less than 100 that comes from a *modular* representation (where the characteristic

* Received by the editors January 21, 1988; accepted for publication July 8, 1988.

† Department of Mathematics, Louisiana State University, Baton Rouge, Louisiana 70803. Present address, AT&T Bell Laboratories, Holmdel, New Jersey 07733.

‡ Mathematical Sciences Research Center, AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

of the field divides the order of the group). Furthermore, in contrast to many of the papers mentioned, we do not use the *regular* representation of the group. Another distinguishing feature of our approach is the relatively large number of invariant subspaces that occur, increasing the chance that one of them is good!

**2. The new code.** Let $G$ be the automorphism group of the four-dimensional cube, a group of structure $2^4.S_4$ and order $2^4 \cdot 4! = 384$. This group permutes the 32 edges of the cube, which we label as in Fig. 1. Let $V$ be a 32-dimensional vector space over $GF(2)$ with basis that is in one-to-one correspondence with the edges, so that $G$ acts on $V$. A typical vector $v \in V$ has the form $v = (v_1, \cdots, v_{32})$, $v_i = 0$ or 1, with coordinates corresponding to the labels in Fig. 1. We write these vectors in hexadecimal notation, with $\mathsf{0} = 0000, \cdots, \mathsf{9} = 1001, \mathsf{A} = 1010, \cdots, \mathsf{F} = 1111$. We may also identify $v$ with the corresponding set of edges.

Any set of vectors $u, v, \cdots \in V$ generates a binary linear code of length 32, denoted by $\langle u, v, \cdots \rangle$, namely the modulo-2 span of the union of the orbits of $u, v, \cdots$ under $G$. These codes are the $G$-invariant subspaces of $V$. A code or subspace $\langle u \rangle$ with a single generator is called *cyclic*, following [18, p. 52]. (This is the appropriate generalization of the standard term from coding theory.)

We denote the $G$-invariant codes of dimension $k$ by $C_k = C_k^{(0)}, C_k^{(1)}, \cdots$, and when they are cyclic we denote corresponding generating vectors by $u_k = u_k^{(0)}, u_k^{(1)}, \cdots$. The labels are chosen so that, for $k \neq 16$, $C_k^{(i)}$ and $C_{32-k}^{(i)}$ are dual codes. Also $C_{16}^{(2i)}$ and $C_{16}^{(2i+1)}$ are duals $(0 \leqq i \leqq 2)$. We shall make use of the particular generators $u_k^{(i)}$ shown in Table 1. Some generators that represent geometrically interesting configurations in the cube are displayed in Fig. 2.

The code $C_{17}$ is the most interesting, and we summarize its properties in the following theorem.

THEOREM 1. *The code* $C_{17} = \langle u_{13}, u_{14} \rangle$ *is a* [32, 17, 8] *binary code, with generator matrix as shown in Fig.* 3(a). (*An alternation definition is given in* § 3.) *This code has the following weight distribution*:

| $i$ | 0 | 8 | 10 | 12 | 14 | 16 |
|---|---|---|---|---|---|---|
| $A_i$ | 1 | 908 | 3328 | 14784 | 27392 | 38246 |

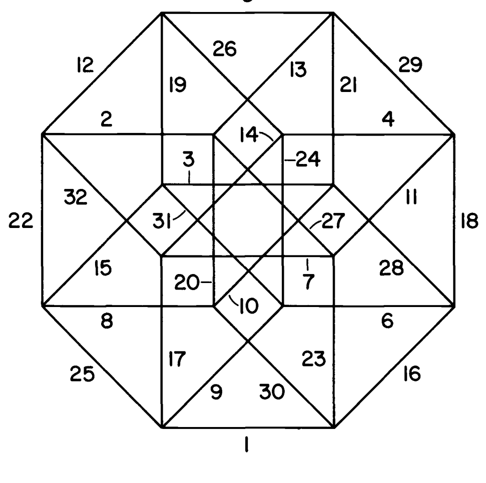


FIG. 1. *Four-dimensional cube with the* 32 *edges labeled.*

TABLE 1
*Generating vectors $u_k^{(i)}$ for selected code $C_k^{(i)}$ (in hexadecimal).*

| | | | |
|---|---|---|---|
| $u_1$ | FFFFFFFF | $u_9^{(9)}$ | 111111EE |
| $u_3$ | 00FF00FF | $u_9^{(10)}$ | 0F0F5A5A |
| $u_4$ | FF000000 | $u_9^{(11)}$ | 0F0F5AA5 |
| $u_5$ | F0F0F0F0 | $u_{11}$ | 00335A69 |
| $u_5^{(1)}$ | CC99CC99 | $u_{11}^{(1)}$ | 11117822 |
| $u_6$ | 55AA5555 | $u_{11}^{(2)}$ | 003C3C5A |
| $u_6^{(1)}$ | 00665533 | $u_{11}^{(3)}$ | 111E111E |
| $u_6^{(2)}$ | 006655CC | $u_{11}^{(4)}$ | 11224B78 |
| $u_7$ | 0F695A3C | $u_{11}^{(6)}$ | 00005A3C |
| $u_7^{(1)}$ | 3C693C69 | $u_{13}$ | AAA50000 |
| $u_7^{(2)}$ | 000F00F0 | $u_{13}^{(1)}$ | 1111444B |
| $u_7^{(3)}$ | 33663C69 | $u_{13}^{(2)}$ | 030648E7 |
| $u_7^{(4)}$ | 1E2D4B78 | $u_{13}^{(3)}$ | 03068481 |
| $u_7^{(5)}$ | 00335566 | $u_{13}^{(4)}$ | 11111E1E |
| $u_7^{(7)}$ | 1E2D4B87 | $u_{13}^{(5)}$ | 03060306 |
| $u_7^{(8)}$ | 11224477 | $u_{14}^{(1)}$ | 03770605 |
| $u_7^{(9)}$ | 88112244 | $u_{14}^{(2)}$ | 0F184184 |
| $u_7^{(10)}$ | 0F3C3C5A | $u_{14}^{(3)}$ | 03090CCA |
| $u_7^{(11)}$ | 0F3C3CA5 | $u_{16}$ | 03091242 |
| $u_9$ | 0F0F3C69 | $u_{16}^{(2)}$ | 03116050 |
| $u_9^{(1)}$ | 0F3C5A69 | $u_{16}^{(4)}$ | 00174184 |
| $u_9^{(2)}$ | 003C5569 | $u_{27}$ | 88840000 |
| $u_9^{(3)}$ | 33693369 | $u_{28}$ | 00008200 |
| $u_9^{(4)}$ | 1E1E1E1E | $u_{29}$ | 80808040 |
| $u_9^{(7)}$ | 1E1E1EE1 | $u_{31}$ | 08100000 |
| $u_9^{(8)}$ | 11111111 | | |

*with $A_{32-i} = A_i$, and $G$ is its full automorphism group. The covering radius of $C_{17}$ is* 6, *a typical deep hole being* **00001117** (*in hexadecimal*). *The dual code is $C_{15} = \langle u_{15} \rangle$, a* [32, 15, 8] *code with generator matrix as shown in Fig.* 3(b), *and has the following weight distribution*:

| $i$ | 0 | 8 | 10 | 12 | 14 | 16 |
|---|---|---|---|---|---|---|
| $A_i$ | 1 | 124 | 1152 | 3584 | 6016 | 11014 |

*with $A_{32-i} = A_i$. All $G$-invariant subcodes of $C_{17}$ and $C_{15}$ are as shown in Figs.* 4 *and* 5; *in particular $C_{17}$ and $C_{15}$ intersect in the* [32, 9, 8] *code $C_9^{(5)}$. The double circles in Figs.* 4 *and* 5 *show all the cyclic modules in these diagrams*; *$C_{17}$ itself is not cyclic.*

*Remarks.* (i) The dual lattice to Fig. 5 gives all the codes containing $C_{17}$.

(ii) The best way to remember these codes is to notice that the generator $u_{15}$ for the dual $C_{15}$ resembles two umbrellas, one of which has lost its fabric (see Fig. 2). This vector is stabilized by a subgroup of $G$ of order six.

(iii) In Table 1 we give more than enough generators to enable any of the codes in Figs. 4 and 5 or their duals to be reconstructed. (The Bensen and Conway [3] notion of reduced lattice of submodules was helpful in preparing Table 1.) For completeness we note that $G$ itself is generated by the following permutations:

$$(1,15,17,8,9,22)(2,16,19,7,10,24)(3,14,20,6,12,23)$$

$$(4,13,18,5,11,21)(26,27,28)(30,31,32)$$

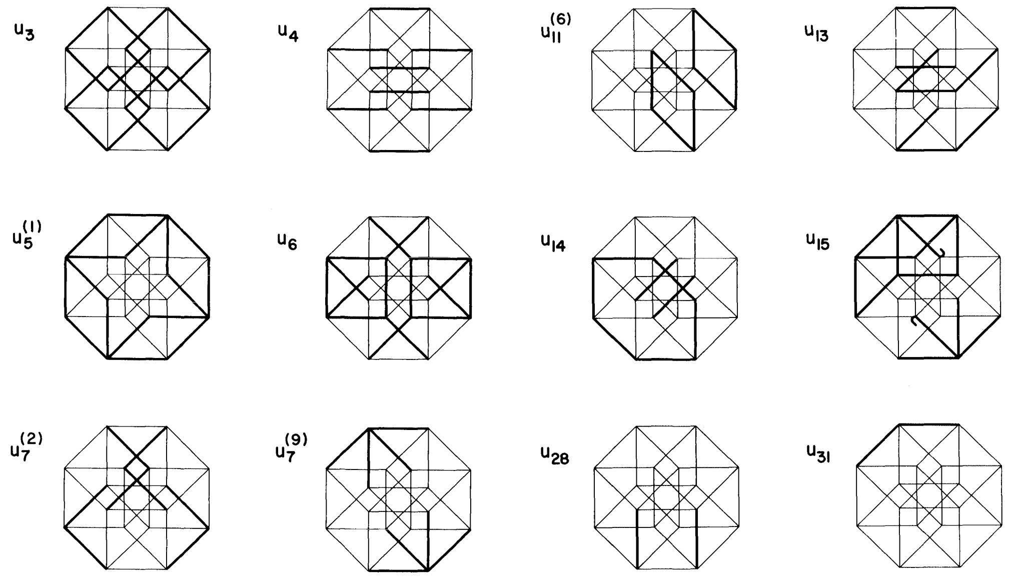


FIG. 2. *Subsets of edges corresponding to selected generating vectors.*

(a)

```
10101010101001010000000000000000
01101010111000010000011010100000
00110000000100010000011000000101
00010001000001100000010100110000
00001010011001010100010000000110
00000110001000010010010000001100
00000011000100010110111110100000
00000000101010101010010100000000
00000000010101011010010100000000
00000000001111000101101001100110
00000000000101110001010000100001
00000000000011110110011000110011
00000000000000001111111100000000
00000000000000001010101010111010
00000000000000000110011011001100
00000000000000000011110000111100
00000000000000000000000011111111
```

(b)

```
11101000000110110010111001000100
01000100111010000001101100101110
00100010010000101101011100101110
00010001101100101110010010001011
00001001000011000110111100111111
00000110101010010011010111001111
00000011000010010011111101010011
00000000100110011010101000110011
00000000011001101010101011001100
00000000001111001001011000110011
00000000000011110101010100001111
00000000000000001111111100000000
00000000000000001011010001111100
00000000000000000110011011001100
00000000000000000000000011111111
```

FIG. 3. *Generator matrices for codes* (a) $C_{17}$ *and* (b) $C_{15}$.

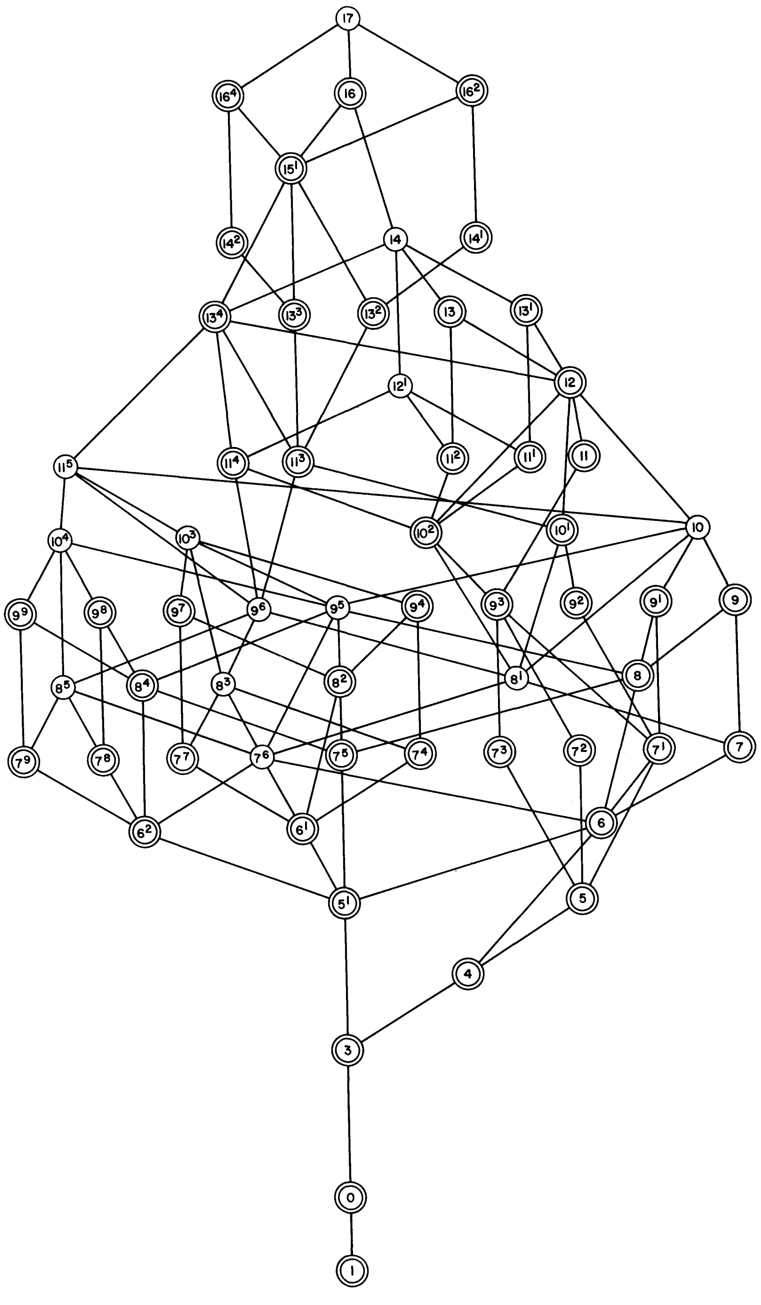


FIG. 4. *Complete lattice of G-invariant subcodes of $C_{17}$. The code $C_k^{(i)}$ is abbreviated $k^i$ in Figs.* 4 *and* 5. *Cyclic modules (with one generator) are indicated by double circles.*

and

$$(1,9,17,25)(2,10,18,26)(3,11,19,27)(4,12,20,28)$$
$$(5,13,21,29)(6,14,22,30)(7,15,23,31)(8,16,24,32).$$

(iv) The following list identifies, from the set of codes mentioned in Figs. 4 and 5 and their duals, all those that have minimal distance $d \geqq 6$:

$d=6$: $C_6, C_{16}^{(3)}, C_{17}^{(1)}, C_{18}, C_{18}^{(2)}$;

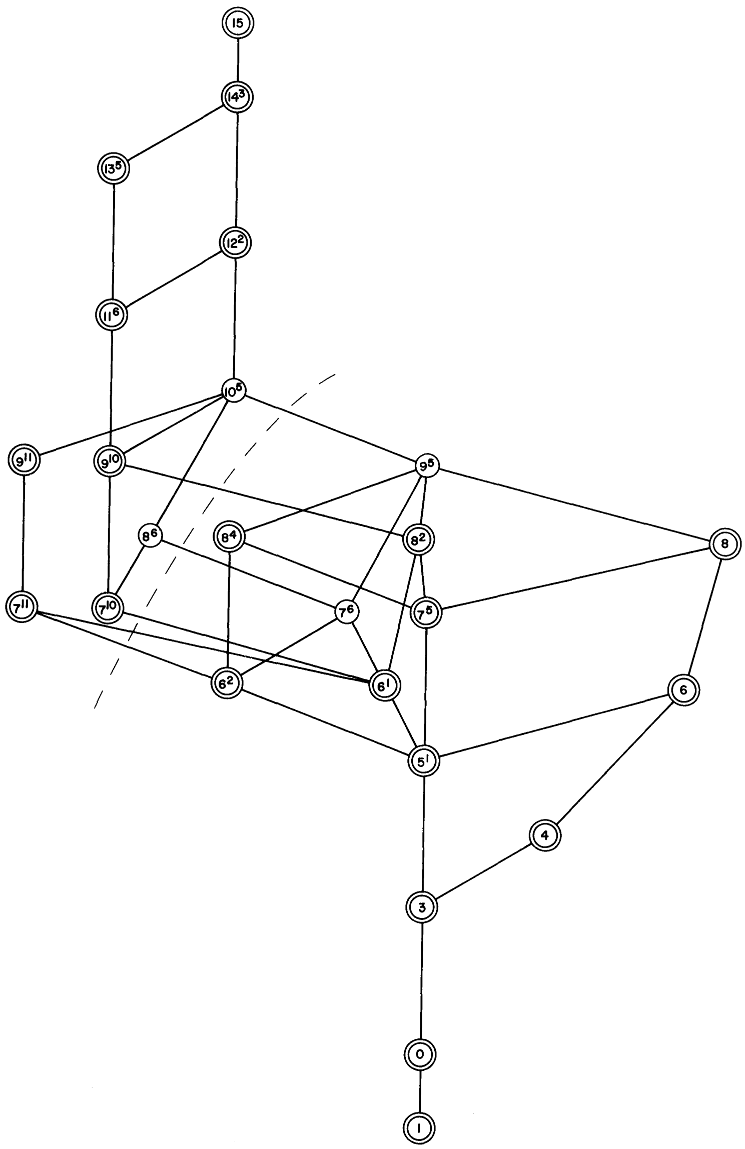


FIG. 5. *Complete lattice of G-invariant subcodes of $C_{15}$.*

$d=8$: $C_4, C_5, C_7^{(i)}(i=0,\cdots,3,6,9), C_8^{(i)}(i=0,\cdots,6), C_9^{(i)}(i=0,\cdots,11),$
$C_{10}^{(i)}(i=0,\cdots,5), C_{11}^{(i)}(i=0,\cdots,6), C_{12}^{(i)}(i=0,1,2), C_{13}^{(i)}(i=0,\cdots,5),$
$C_{14}^{(i)}(i=0,\cdots,3), C_{15}, C_{15}^{(1)}, C_{16}^{(i)}(i=0,\cdots,5), C_{17};$

$d=12$: $C_6^{(1)}, C_6^{(2)}, C_7^{(i)}(i=4,5,7,8,10,11);$

$d=16$: $C_3, C_5^{(1)};$

$d=32$: $C_1$.

(v) A dense 32-dimensional lattice sphere packing may be obtained from $C_{17}$ by applying Construction D of [2]. This packing (see [17, p. 235]) has center density $\delta =$

2 and each sphere touches 249,280 others, and is the second densest packing known in this dimension. (Quebbemann's 32-dimensional lattice ([30], [31], [17, p. 220]) has $\delta = 2.566 \cdots$ and each sphere touches 261,120 others.)

The group $G = 2^4.S_4$, like $S_4$, has just two conjugacy classes of elements of odd order, and so, again like $S_4$, has just two absolutely irreducible representations over $GF(2)$ (cf. [18, p. 58]). These are the trivial one-dimensional representation and the two-dimensional representation by the following matrices of $GL_2(2)$:

$$(1) \qquad \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \right\}.$$

THEOREM 2. (a) *Every composition series of $V$ begins and ends*:

$$\{0\} = C_0 \subset C_1 \subset \cdots \subset C_{31} \subset C_{32} = V,$$

*where $C_1 = \{0^{32}, 1^{32}\}$ and $C_{31}$ consists of all even-weight vectors. In particular, every nontrivial $G$-invariant code is even, contains $1^{32}$, and its weight distribution satisfies $A_i = A_{32-i}$.*

(b) *One composition series for $V$ is*

$$\begin{aligned} \{0\} &= C_0 \subset C_1 \subset C_3 \subset C_4 \subset C_6 \subset C_7 \subset C_9 \subset C_{10} \\ &\subset C_{12} \subset C_{13} \subset C_{14} \subset C_{16} \subset C_{17} \subset C_{18}^{(3)} \subset C_{20}^{(2)} \subset C_{22}^{(5)} \\ &\subset C_{23}^{(5)} \subset C_{24} \subset C_{26} \subset C_{28} \subset C_{29} \subset C_{31} \subset C_{32} = V. \end{aligned}$$

(c) *The composition factors for $V$ are $1^{12}2^{10}$.*

Before proving these theorems we describe what we think is the full list of invariant subspaces.

CONJECTURE. (a) The complete list of $G$-invariant subcodes of $V$ consists of 373 codes, whose dimensions $k$ are as follows:

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| # | 1 | 0 | 1 | 1 | 2 | 3 | 14 | 16 |

| $k$ | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|
| # | 20 | 16 | 19 | 16 | 17 | 22 | 22 | 31 |

(The number of dimension $32 - k$ is equal to the number of dimension $k$.)

(b) The code $C_{17}$ is the unique $G$-invariant code of minimal distance $d \geqq 8$ and dimension $k \geqq 17$. The largest $G$-invariant codes of minimal distances 4, 6, 12, 16 have dimensions 25, 18, 8, 5, respectively (and are not especially good; cf. Verhoeff [33]).

(c) There are nine self-dual codes, all with minimal distance $d = 2$ or 4 (e.g., the vectors 0 0 0 0 0 0 1 1, 0 0 0 0 0 0 0 F generate self-dual codes with $d = 2, 4$, respectively). The nontrivial Reed–Muller, extended Hamming, and extended quadratic-residue codes of length 32 are not $G$-invariant codes.

*Remark.* The 373 codes described in (a) (and in Figs. 4 and 5) are only claimed to be distinct, not necessarily inequivalent. But usually distinct $G$-invariant codes are inequivalent. More precisely, if $C$ and $C'$ are equivalent codes (implying that there is a permutation $\pi \in S_{32}$ with $C^\pi = C'$) such that $\mathrm{Aut}(C) = \mathrm{Aut}(C') = G$, then $C = C'$. For $\mathrm{Aut}(C') = \pi\, \mathrm{Aut}(C)\pi^{-1} = \mathrm{Aut}(C) = G$, implying that $\pi$ is in the normalizer of $G$ in $S_{32}$. But $G$ is equal to its normalizer, so $\pi \in G$, and $C = C'$.

*Proof of Theorem* 1. The assertions about the dimension, weight distribution, covering radius, and dual code are routine computer verifications.

By definition, $\mathrm{Aut}(C_{17}) \supseteq G$. To prove equality, we first examined (by computer) the weight distributions of the nonlinear subcode formed by the 908 codewords of weight

8. There are exactly four weight 8 codewords with weight distribution $A_0 = 1, A_8 = 180$, $A_{17} = 544, A_{16} = 183$, namely the vectors **FF000000**, **00FF0000**, **0000FF00**, **000000FF**. (These are supported on the four classes of eight parallel edges of the cube, see Fig. 1.) Thus the division of the 32 coordinates into these four blocks of eight is canonical. The group $G$ induces all 4! permutations of the four blocks.

There are exactly 28 codewords meeting the blocks $4 + 4 + 0 + 0$, and these have the form $(u, u, 0, 0)$ and $(u, \bar{u}, 0, 0)$, where $u$ is a weight 4 word in an $[8, 4, 4]$ Hamming code $\mathscr{H}_8$. The automorphism group of $\mathscr{H}_8$ has structure $2^3.L_2(7)$ [2, p. 399], and the permutations induced by $G$ on the first block yield exactly the $2^3$ part of this group. $G$ also contains the permutation $(9, 10)(11, 12) \cdots (31, 32)$, fixing the blocks and fixing every point of the first block. Any permutation of $C_{17}$ not in $G$ can then be assumed to fix the blocks, and to act as an element of $L_2(7)$ inside each block. We verified by computer that all such permutations are already in $G$. Thus $\operatorname{Aut}(C_{17}) = G$.

The assertion that Figs. 4 and 5 show all $G$-invariant subcodes of $C_{17}$ and $C_{15}$ was proved as follows. We first established what we believe is a complete list of all $G$-invariant subcodes of $V$. There are 373 codes, as described above. (This list was constructed by a variety of techniques: repeatedly taking joins, intersections, and duals; constructing a generator matrix for each code and finding the cyclic module generated by each row; finding the cyclic modules generated by all vectors of selected codes; and other ad hoc methods.) The list was checked to be closed under the operations of taking joins, intersections, and duals. We examined the cyclic codes generated by every vector of $C_{17}$ and $C_{15}$, and verified that these are on the list. This proves the assertion.

*Proof of Theorem* 2. (a) From the remarks preceding the theorem we know that the composition factors are all 1 or 2. Suppose a composition series begins $C_0 \subset C_2 \subset \cdots$, where $C_2$ is a two-dimensional code generated by vectors $u$, $v$ and affording the two-dimensional representation (1). Then every $g \in G$ sends $u$ to $u$, $v$ or $u + v$, and all three occur. Since $G$ is transitive, $|u \cap \bar{v}| + |u \cap v| + |\bar{u} \cap v| = 32$. Since $u$ can be mapped to $v$, $|u \cap \bar{v}| = |\bar{u} \cap v|$, and similarly $|u \cap \bar{v}| = |u \cap v|$, so the three sets are equal in size and $3|u \cap v| = 32$, which is impossible. The assertion $\cdots C_{31} \subset C_{32}$ follows by duality.

$$\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
a & a & 0 & 0 \\
b & b & 0 & 0 \\
c & c & 0 & 0 \\
a & 0 & a & 0 \\
b & 0 & b & 0 \\
c & 0 & c & 0 \\
a & 0 & 0 & a \\
b & 0 & 0 & b \\
c & 0 & 0 & c \\
0 & w & w & w \\
x & 0 & x & x \\
y & y & 0 & y \\
z & z & z & 0
\end{bmatrix}$$

FIG. 6. *Alternative generator matrix for* $C_{17}$ ($0 = 00000000$, $1 = 11111111$).

(b), (c) The computer was used to verify that all the composition factors of 2 in the given series are irreducible.

**3. An alternative construction.** The [32, 17, 8] code $C_{17}$ described in Theorem 1 was in fact first found by the following construction. This provides an alternative description, and may be of independent interest.

Let $\mathscr{H}_8$ and $\mathscr{H}'_8$ be two versions of the [8, 4, 4] Hamming code that intersect only in $\{0^8, 1^8\}$. (For example, take the point-code and line-code shown in [17, Fig. 11.27].) Choose independent vectors $a, b, c \in \mathscr{H}_8$ that span $\mathscr{H}_8/\{0^8, 1^8\}$, and vectors $w, x, y, z \in \mathscr{H}'_8$ that span $\mathscr{H}'_8/\{0^8, 1^8\}$ and satisfy $w + x + y + z = 0$. (For example, $a = 10101001$, $b = 10011100$, $c = 10000111$, $w = 11001100$, $x = 10101010$, $y = 11110000$, $z = 10010110$.) Then Fig. 6 generates a code equivalent to $C_{17}$. (It is not difficult to find an isomorphism onto the earlier version. The first four rows of Fig. 6 are the four special codewords mentioned in the proof of Theorem 1.)

**Acknowledgments.** We are grateful to John Conway and Walter Feit for some very helpful suggestions.

**Note added in proof.** Gerhard J. A. Schneider of the University of Essen has verified that Conjecture (a) is correct, using the CAYLEY computer system. There are indeed exactly 373 $G$-invariant subcodes. (Personal communication, June 10, 1988.)

## REFERENCES

[1] R. Ahlswede and G. Dueck, *Good codes can be produced by a few permutations*, IEEE Trans. Inform. Theory, 27 (1981), pp. 398–408.

[2] E. S. Barnes and N. J. A. Sloane, *New lattice packings of spheres*, Canad. J. Math., 35 (1983), pp. 117–130.

[3] D. J. Benson and J. H. Conway, *Diagrams for modular lattices*, J. Pure Appl. Algebra, 37 (1985), pp. 111–116.

[4] S. D. Berman, *On the theory of group codes*, Kibernetika, 3, no. 1, (1967), pp. 31–39. (In Russian.) Cybernetics, 3, no. 1, (1967), pp. 25–31. (In English.)

[5] ———, *Semisimple cyclic and Abelian codes* II, Kibernetika, 3, no. 3, (1967), pp. 21–30. (In Russian.) Cybernetics, 3, no. 3, (1967), pp. 17–23. (In English.)

[6] P. Bhattacharya, *On a class of Abelian codes*, Inform. Sci., 33 (1984), pp. 173–179.

[7] P. L. H. Brooke, *Tables of codes associated with certain finite simple groups*, Ph.D. Dissertation, University of Cambridge, Cambridge, UK, June 1984.

[8] ———, *On matrix representations and codes associated with the simple group of order* 25920, J. Algebra, 91 (1984), pp. 536–566.

[9] ———, *On the Steiner system* $S(2, 4, 28)$ *and codes associated with the simple group of order* 6048, J. Algebra, 97 (1985), pp. 376–406.

[10] A. R. Calderbank and D. B. Wales, *A global code invariant under the Higman-Sims group*, J. Algebra, 75 (1982), pp. 233–260.

[11] P. Camion, *Abelian Codes*, Report 1059 Mathematics Research Center, University of Wisconsin, 1970.

[12] ———, *Etude des codes binaires Abelians modulaires autoduaux de petites longuers*, Rev. CETHEDEC, 1979, no. 2, pp. 3–24.

[13] P. Charpin, *The extended Reed-Solomon codes considered as ideals of a modular algebra*, Ann. Discrete Math., 17 (1983), pp. 171–176.

[14] ———, *A description of some extended cyclic codes with application to Reed–Solomon codes*, Discrete Math., 56 (1985), pp. 117–124.

[15] J. H. Conway, S. J. Lomonaco, Jr., and N. J. A. Sloane, *A* [45, 13] *code with minimal distance* 16, Discrete Math., to appear.

[16] J. H. Conway and V. Pless, *On the enumeration of self-dual codes*, J. Combin. Theory Ser. A, 28 (1980), pp. 26–53.

[17] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1988.

[18] C. W. Curtis and I. Reiner, *Representation Theory of Finite Groups and Associative Algebras*, John Wiley, New York, 1962.

[19] P. Delsarte, *Automorphisms of Abelian codes*, Philips Res. Reports, 25 (1970), pp. 389–402.

[20] J. M. JENSEN, *The concatenated structure of cyclic and Abelian codes*, IEEE Trans. Inform. Theory, 31 (1985), pp. 788–793.

[21] M. KLEMM, *Kennzeichnung der erweiterten Quadrate-codes durch ihre $PSL(2, q)$-Zulässigkeit*, Commun. Algebra, 11 (1983), pp. 2051–2068.

[22] P. LANDROCK AND O. MANZ, *Classical codes as ideals in group algebras*, preprint.

[23] R. A. LIEBLER, *On codes in the natural representations of the symmetric group*, in Combinatorics, Representation Theory, and Statistical Methods in Groups, Marcel Dekker, New York, 1980.

[24] C. L. MALLOWS AND N. J. A. SLOANE, *An upper bound for self-dual codes*, Inform. and Control, 22 (1973), pp. 188–200.

[25] F. J. MACWILLIAMS, *Codes and ideals in group algebras*, in Combinatorial Mathematics and Its Applications, R. C. Bose and T. A. Dowling, eds., University North Carolina Press, Chapel Hill, NC, 1969, Chap. 18.

[26] ———, *Binary codes which are ideals in the group algebra of an Abelian group*, Bell System Tech. J., 49 (1970), pp. 987–1011.

[27] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1979.

[28] R. A. PARKER, *The computation of modular characters (the meat-axe)*, in Computational Group Theory, M. D. Atkinson, ed., Academic Press, New York, 1984, pp. 267–274.

[29] J. N. PIERCE, *Limit distribution of the minimum distance of random linear codes*, IEEE Trans. Inform. Theory, 13 (1967), pp. 595–599.

[30] H.-G. QUEBBEMANN, *A construction of integral lattices*, Mathematika, 31 (1984), pp. 137–140.

[31] ———, *Lattices with theta-functions for $G(\sqrt{2})$ and linear codes*, J. Algebra, 105 (1987), pp. 443–450.

[32] P. RABIZZONI, *Images binaires d'ideaux principaux d'une algèbre de groupe*, Rev. CETHEDEC, 1981, no. 2.

[33] T. VERHOEFF, *An updated table of minimum-distance bounds for binary linear codes*, IEEE Trans. Inform. Theory, 33 (1987), pp. 665–680.

[34] H. N. WARD, *Divisible codes*, Arch. Math., 36 (1981), pp. 485–494.

[35] J. WOLFMANN, *A group algebra construction of binary even self-dual codes*, Discrete Math., 65 (1987), pp. 81–89.

[36] B. M. ZLOTNIK, *Doubly transitive groups of type $p^m(p^m - 1)$ and maximal nonbinary codes generated by them*, Kibernetika, 19, no. 3, (1983), pp. 16–20. (In Russian.) Cybernetics, 19 (1983), pp. 309–316. (In English.)

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 38-47, February 1989

# GRAPH BIPARTIZATION AND VIA MINIMIZATION*

HYEONG-AH CHOI†, KAZUO NAKAJIMA‡, AND CHONG S. RIM§

**Abstract.** The vertex- (respectively, edge-) deletion graph bipartization problem is the problem of deleting a set of vertices (respectively, edges) from a graph so as to make the remaining graph bipartite. This paper first shows that the vertex-deletion graph bipartization problem has a solution of size $k$ or less if and only if the edge-deletion graph bipartization problem has a solution of size $k$ or less, when the maximum vertex degree is limited to three. This immediately implies that (1) the vertex-deletion graph bipartization problem is NP-complete for cubic graphs, and (2) the minimum vertex-deletion graph bipartization problem is solvable in polynomial time for planar graphs when the maximum vertex degree is limited to three. It is then proved that the vertex-deletion graph bipartization problem is NP-complete for planar graphs when the maximum vertex degree exceeds three. Using this result, it is finally shown that the via minimization problem, which arises in the design of integrated circuits and printed circuit boards, is NP-complete even when the maximum "junction" degree is limited to four.

**Key words.** computational complexity, maximum bipartite subgraphs, NP-completeness, via minimization

**AMS(MOS) subject classifications.** 68C25, 68E10

**1. Introduction.** Let $G = (V, E)$ be a graph. We denote by $d_G(v)$ the degree of vertex $v \in V$ in $G$ and let $\Delta(G) = \max_{v \in V}\{d_G(v)\}$. $G$ is called a *bipartite graph* if $V$ can be partitioned into two nonempty subsets $V_1$ and $V_2$ such that $V_1 \cap V_2 = \phi$ and no two vertices in the same subset are adjacent. $V_1$ and $V_2$ are called a *bipartition* of $G$. It is well known [4] that the graph $G$ is bipartite if and only if there is no cycle of odd length in $G$.

For a subset $V' \subset V$ of $G = (V, E)$, the graph obtained by deleting from $G$ all vertices in $V'$ and all the edges incident upon them is called a *vertex-deleted subgraph* of $G$ and is denoted by $G^v(V - V')$; that is, $G^v(V - V') = (V - V', E(V - V'))$ where $E(V - V') = \{(u, v) \in E \mid u, v \in V - V'\}$. Similarly, for a subset $E' \subset E$ of $G = (V, E)$, an *edge-deleted subgraph* of $G$, denoted by $G^e(E - E')$, is obtained by deleting all edges in $E'$ from $G$; that is, $G^e(E - E') = (V, E - E')$.

Given a graph $G = (V, E)$ and an integer $k \geqq 0$, the *vertex-* (respectively, *edge-*) *deletion graph bipartization problem*, (VDB) (respectively, EDB), is the problem of finding a set of $k$ or fewer vertices $V' \subset V$ (respectively, edges $E' \subset E$) in $G$ such that the subgraph $G^v(V - V')$ (respectively, $G^e(E - E')$) is bipartite, or equivalently, free of odd length cycles. The *minimum vertex-* (respectively, *edge-*) *deletion graph bipartization problem*, (MVDB) (respectively, MEDB), is the problem of finding such a vertex (respectively, edge) set of minimum cardinality.

Garey, Johnson, and Stockmeyer [7] and Yannakakis [17] proved that the EDB problem is NP-complete even if $G = (V, E)$ is a cubic graph, i.e., $d_G(v) = 3$ for every

* Received by the editors December 7, 1987; accepted for publication (in revised form) July 28, 1988.

† Department of Electrical Engineering and Computer Science, George Washington University, Washington, D.C. 20052. This work was performed while the author was with the Department of Computer Science, Michigan State University, East Lansing, Michigan 48824 and was supported by the Division of Engineering Research of Michigan State University.

‡ Electrical Engineering Department, Institute for Advanced Computer Studies and Systems Research Center, University of Maryland, College Park, Maryland 20742. This research was supported in part by National Science Foundation grants MIP-84-51510 and CDR-85-00108 and by a grant from AT&T.

§ Electrical Engineering Department and Systems Research Center, University of Maryland, College Park, Maryland 20742. This work was supported in part by National Science Foundation grants MIP-84-51510 and CDR-00108 and by a grant from AT&T.

$v \in V$. On the other hand, Hadlock [8], [1] showed that the MEDB problem is solvable in $O(|V|^3)$ time if the graph is planar. As for the VDB problem, the combined results of Garey and Johnson [6] and Krishnamoorthy and Deo [11] imply that it is NP-complete even if $G$ is planar and $\Delta(G) = 6$. Given a graph $G = (V, E)$, the *line graph*, denoted by $G_l = (V_l, E_l)$, of $G$ is defined as follows: There is a one-to-one correspondence between $V_l$ and $E$. If two edges in $E$ are incident upon the same vertex in $G$, then there is an edge in $E_l$ which connects the two corresponding vertices in $V_l$. We can easily show by constructing the line graph of a cubic graph that the VDB problem is NP-complete for a general graph $G$ with $\Delta(G) = 4$.

The first goal of this paper is to present complete complexity results for the VDB problem. We arrive at this by first establishing a relationship between the VDB and EDB problems for a graph $G = (V, E)$ with $\Delta(G) \leqq 3$. Namely, we prove that the VDB problem has a solution $V' \subset V$ with $|V'| \leqq k$ if and only if the EDB problem has a solution $E' \subset E$ with $|E'| \leqq k$. Since the EDB problem is NP-complete for cubic graphs [7], [17], this immediately implies that the VDB problem is NP-complete for cubic graphs. Furthermore, since the MEDB problem is solvable in $O(|V|^3)$ time if $G = (V, E)$ is planar [8], [1], this relationship also implies that the MVDB problem is solvable in $O(|V|^3)$ time for the case when $G$ is planar and $\Delta(G) \leqq 3$. It should be noted further that if $\Delta(G) \leqq 2$, the graph $G$ is always planar and hence the VDB problem is solvable in polynomial time, and in fact, in $O(|V|)$ time. Finally, we prove that the VDB problem becomes NP-complete for a planar graph $G$ if $\Delta(G) = 4$. We give a polynomial transformation from the Planar 3-Satisfiability problem [12] to this problem. The complete complexity results for the VDB problem are summarized in Table 1.

The second goal of this paper is to show the NP-completeness of the *via minimization problem* which arises in the design of integrated circuits and printed circuit boards. A *via* is a contact cut or hole which is needed to electrically connect wire segments of the same signal net when they are assigned to different layers. Given a set of wire segments and two layers for routing, the problem is to assign the segments to one of the layers so as to minimize the number of vias required.

In 1971, Hashimoto and Stevens [9] first considered this problem for the case of grid-based layouts with the maximum "junction" degree being limited to two, where a *grid-based layout* is a layout in which all wire segments are placed in parallel to one of the two perpendicular axes, and "*junction*" *degree* is defined to be the number of wire segments which meet at a single point and which are to be electrically connected. In 1980, Kajitani [10] showed that the problem is solvable in polynomial time for this particular case. Later, Chen, Kajitani, and Chan [2] and Pinter [16] extended its polynomial solvability for grid-based layouts to the case of the maximum junction degree three. Quite recently, Molitor [13] and Naclerio, Masuda, and Nakajima [14] showed that the via minimization problem is solvable in polynomial time even for general layouts if the maximum junction degree is limited to three.

TABLE 1
*Complexity results for the* VDB *problem.*

| | $\Delta(G)$ | | |
|---|---|---|---|
| $G$ | $\leq 2$ | $=3$ | $\geq 4$ |
| General | $O(\|V\|)$ | NPC | NPC |
| Planar | $O(\|V\|)$ | $O(\|V\|^3)$ | NPC |

On the other hand, Naclerio, Masuda, and Nakajima [15] showed that the via minimization decision problem is NP-complete if the maximum junction degree exceeds five. Thus, the cases of the maximum junction degrees four and five have been left as open problems. In this paper, using the NP-completeness result for the VDB problem for planar graphs, we prove that the via minimization decision problem is NP-complete even when the maximum junction degree is limited to four.

In § 2, we first show that the MVDB problem for a general graph $G = (V, E)$ is easily solvable in $O(|V|)$ time if $\Delta(G) \leqq 2$. We then prove that if $\Delta(G) = 3$, the VDB problem has a solution of size $k$ or less if and only if the EDB problem has a solution of size $k$ or less. As corollaries of this result, we show that (1) the VDB problem is NP-complete even for cubic graphs, and (2) the MVDB problem is solvable in $O(|V|^3)$ time when $G$ is planar and $\Delta(G) = 3$. Finally, using a transformation from the Planar 3-Satisfiability problem [12], we prove that the VDB problem becomes NP-complete for a planar graph $G$ if $\Delta(G) = 4$. In § 3, we consider the via minimization problem for two layers. We show how the VDB problem for planar graphs can be transformed to this problem. This leads us to prove that the via minimization decision problem is NP-complete even when the maximum junction degree is limited to four. Section 4 concludes the paper with some further comments on the via minimization problem.

**2. Graph bipartization.** In this section, we investigate the computational complexity of the VDB problem from the point of view of vertex degree constraints. We first show that there is a close relationship between the VDB and EDB problems for a graph $G = (V, E)$ with $\Delta(G) \leqq 3$. If $\Delta(G) \leqq 2$, each connected component of the graph $G$ is either a single vertex, a path, or a cycle. Therefore, the MVDB and MEDB problems can be solved by first finding all cycles of odd length and then selecting an arbitrary vertex and edge, respectively, from each such cycle. This can easily be done in $O(|E|)$ time, which is in fact $O(|V|)$ time, since $G$ is planar in this case. We now consider the case of $\Delta(G) = 3$ and establish a close relationship between the VDB and EDB problems.

THEOREM 1. *For any graph $G = (V, E)$ with $\Delta(G) = 3$, there exists a subset $E' \subset E$ such that $|E'| \leqq k$ and $G^e(E - E')$ is bipartite if and only if there exists a subset $V' \subset V$ such that $|V'| \leqq k$ and $G^v(V - V')$ is bipartite.*

*Proof.* Suppose that there exists a subset $E' \subset E$ with $|E'| \leqq k$ such that $G^e(E - E')$ is bipartite. Let $V' = \{v \in V \mid \textit{for each edge } (u, w) \in E', v \textit{ is either } u \textit{ or } w\}$. It is clear that $|V'| \leqq |E'| \leqq k$ and $G^v(V - V')$ is bipartite.

Conversely, suppose that there exists a subset $V' \subset V$ with $|V'| \leqq k$ such that $G^v(V - V')$ is bipartite. Let $V''$ be a minimal subset of $V'$ such that $G^v(V - V'')$ is still bipartite. Let $X$ and $Y$ be a bipartition of $G^v(V - V'')$. The minimality of $V''$ implies that in $G$, every vertex in $V''$ is adjacent to at least one vertex in $X$ and at least one vertex in $Y$. This is because if vertex $v \in V''$ is not adjacent to any vertices in $X$ or $Y$ or both, $G^v((V - V'') \cup \{v\})$ is bipartite, which contradicts the minimality of $V''$. Note that since $\Delta(G) = 3$, $G^v(V'')$ is a collection of isolated edges and vertices.

We now construct a set of edges $E' \subset E$ such that $G^e(E - E')$ is bipartite. If $v \in V''$ is an isolated vertex in $G^v(V'')$, it is adjacent to exactly two vertices in $X$ or $Y$, say $X$, and exactly one vertex, say $y$ in $Y$. Place $v$ in $Y$, remove the edge $(v, y)$ and put it into $E'$. If $(v, w)$ is an isolated edge in $G^v(V'')$, each of the vertices $v$ and $w$ is adjacent to exactly one vertex in $X$ and exactly one vertex in $Y$. Let $x_1, x_2$ be the vertices in $X$ and $y_1, y_2$ be the vertices in $Y$ such that $(v, x_1), (v, y_1), (w, x_2), (w, y_2) \in E$. Place $v$ into $X$ and $w$ into $Y$, remove the edges $(v, x_1)$ and $(w, y_2)$ and put them into $E'$. Obviously, $G^e(E - E')$ is bipartite and $|E'| = |V''| \leqq |V'| \leqq k$. This completes the proof of the theorem. □

Garey et al. [7] and Yannakakis [17] showed that the EDB problem is NP-complete even if a graph $G = (V, E)$ is a cubic graph, i.e., $d_G(v) = 3$ for every $v \in V$. Thus, we can derive the following result from Theorem 1.

COROLLARY 1. *The* VDB *problem is* NP*-complete even for a cubic graph.*

On the other hand, if the graph $G = (V, E)$ is planar, Hadlock [8], [1] showed that the MEDB problem can be solved in $O(|V|^3)$ time. Therefore, we obtain the following result from Theorem 1.

COROLLARY 2. *For a planar graph $G = (V, E)$ with $\Delta(G) = 3$, the* MVDB *problem is solvable in $O(|V|^3)$ time.*

*Remark* 1. The above discussion suggests a way to solve the MVDB problem for a planar graph $G$ with $\Delta(G) = 3$. That is, we first solve the MEDB problem using Hadlock's approach [8], and then convert its solution to a solution to the MVDB problem. We present a more direct approach to the MVDB problem. This approach can also be used to obtain an approximate solution for the case of $\Delta(G) \geqq 4$.

Let $G = (V, E)$ be a given planar graph. Without loss of generality, we can assume that $G$ is 2*-connected*, that is, for every pair of vertices $u$ and $v$ in $V$, there are at least two vertex disjoint paths from $u$ to $v$.

Let $\tilde{G}$ be a planar embedding of $G$. Since $G$ is 2-connected, $\tilde{G}$ partitions the rest of the plane into a number of connected regions. The closures of those regions are called the *faces* of $\tilde{G}$. Let $F$ be a set of such faces. $F$ includes a special face called an *exterior face* which represents the infinite region outside the embedding $\tilde{G}$. Note that each face in $F$ corresponds to a cycle in $G$, which is called a *fundamental cycle*. We create a new graph $G^n = (V^n, E^n)$, where $V^n = V \cup F$ and $E^n = \{(v, f) \mid v \in V$ *is on the fundamental cycle corresponding to* $f \in F\}$. Then we follow Hadlock [8] and find a pairing of odd degree vertices (or faces) in $F$ such that the total sum of lengths of shortest paths between such pairs is a minimum.

We now consider the case in which $G = (V, E)$ is planar and $\Delta(G) = 4$. We prove that the VDB problem becomes NP-complete in this case.

THEOREM 2. *The* VDB *problem is* NP*-complete for a planar graph even if* $\Delta(G) = 4$.

*Proof.* Since the VDB problem belongs to the class NP, it suffices to show that a known NP-complete problem is transformable in polynomial time to this problem. We start with the following problem which was shown to be NP-complete by Lichtenstein [12].

**Planar 3-Satisfiability (P3SAT)**

**Instance:** A set $U = \{v_i \mid 1 \leqq i \leqq n\}$ of $n$ Boolean variables and a set $C = \{c_j \mid 1 \leqq j \leqq m\}$ of $m$ clauses over $U$ such that each clause $c_j$ contains exactly three variables or their complements. Furthermore, the following graph is planar:

$G_C = (V_C, E_C)$, where

$V_C = \{c_j \mid 1 \leqq j \leqq m\} \cup \{v_i \mid 1 \leqq i \leqq n\}$ and

$E_C = \{(c_j, v_i) \mid v_i \in c_j \text{ or } \bar{v}_i \in c_j\} \cup \{(v_i, v_{i+1}) \mid 1 \leqq i < n\} \cup \{(v_n, v_1)\}$.

**Question:** Is $C$ satisfiable? Namely, is there a truth assignment for $U$ such that each clause in $C$ is true?

Given $U = \{v_i \mid 1 \leqq i \leqq n\}$ and $C = \{c_j \mid 1 \leqq j \leqq m\}$ together with a planar embedding of $G_C = (V_C, E_C)$, we construct a planar graph $G = (V, E)$ in the following way. The graph contains two kinds of components: *clause components* and *variable components.* And each of them is placed in the region in which the corresponding vertex $c_j \in V_C$ or

$v_i \in V_C$ is drawn. For each clause $c_j \in C$ we create a clause component of four vertices $w_j^1, w_j^2, w_j^3, w_j^4$ and one edge $(w_j^1, w_j^4)$ as shown in Fig. 1(a). As will be explained later, each pair of vertices $w_j^l$ and $w_j^{l+1}$ for $l = 1, 2, 3$, will be connected to a pair of vertices of an appropriate variable component. For each variable $v_i \in U$ we construct a variable component of $8n_i$ vertices and $12n_i$ edges, where $n_i$ is the number of times variable $v_i$ or its complement $\bar{v}_i$ appears in $C$. As illustrated in Fig. 1(b), each variable component is made up of $4n_i$ triangles whose bottom edges form a cycle of length $4n_i$. The vertices on this cycle correspond to $v_i$ and $\bar{v}_i$, alternately. Namely, $b_i^{k1}$ and $b_i^{k3}$ correspond to $v_i$ and $b_i^{k2}$ and $b_i^{k4}$ correspond to $\bar{v}_i$ for $k = 1, 2, \cdots, n_i$. Note that a group of four consecutive triangles, more precisely, eight vertices $a_i^{kl}$ and $b_i^{kl}$ for $l = 1, 2, 3, 4$ correspond to variable $v_i$. If $\bar{v}_i \in c_j$ (respectively, $v_i \in c_j$), then the two top vertices $a_i^{k1}$ and $a_i^{k2}$ (respectively, $a_i^{k2}$ and $a_i^{k3}$) for some $k \in \{1, 2, \cdots, n_i\}$ are connected to a pair of vertices $w_j^l$ and $w_j^{l+1}$ for some $l \in \{1, 2, 3\}$ of the clause component corresponding to $c_j$. For example, see Fig. 2. It is easy to see that $|V| = 4m + \sum_{i=1}^n 8n_i = 28m$ and $|E| = 7m + \sum_{i=1}^n 12n_i = 43m$. Therefore, this transformation is done in polynomial time. Furthermore, it is clear that $G$ is planar and $\Delta(G) = 4$.

We now prove that there is a truth assignment for $U$ such that each clause $c_j \in C$ is true if and only if there is a subset $V' \subset V$ of $G = (V, E)$ such that $|V'| = 6m$ and $G^v(V - V')$ is bipartite.

Suppose that there is a subset $V' \subset V$ such that $|V'| = 6m$ and $G^v(V - V')$ is bipartite. Since each variable component contains $4n_i$ triangles, which are odd length cycles, we must delete at least $2n_i$ vertices to break these triangles. This can be done only if every other vertex on the cycle of length $4n_i$ is so chosen. More precisely, we must delete $n_i$ pairs of vertices either $b_i^{k1}$ and $b_i^{k3}$ or $b_i^{k2}$ and $b_i^{k4}$ for $k = 1, 2, \cdots, n_i$, for each $i = 1, 2, \cdots, n$. Since $\sum_{i=1}^n 2n_i = 6m$, no other vertices are deleted. Therefore, we can make a consistent assignment of value *true* or *false* to each variable $v_i$ in the following manner: If vertex $b_i^{12}$ (respectively, $b_i^{13}$) is deleted from the variable component corresponding to $v_i$, assign *false* (respectively, *true*) to $v_i$, for $i = 1, 2, \cdots, n$.

Note that all four vertices in each clause component belong to a cycle of length 13 which connects those four vertices and three vertices, one from each of the three corre-

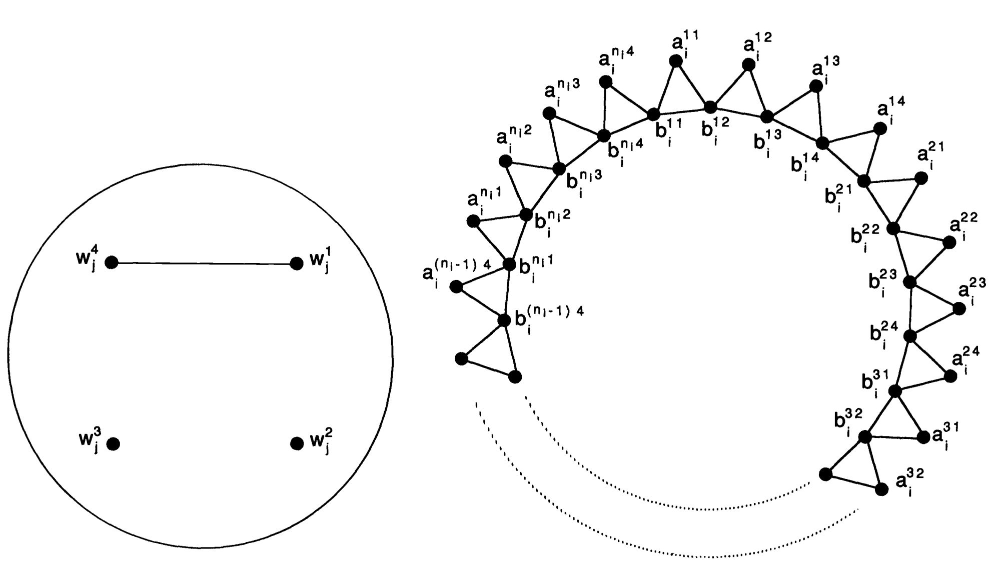


FIG. 1. *Clause and variable components.* (a) *Clause component.* (b) *Variable component.*

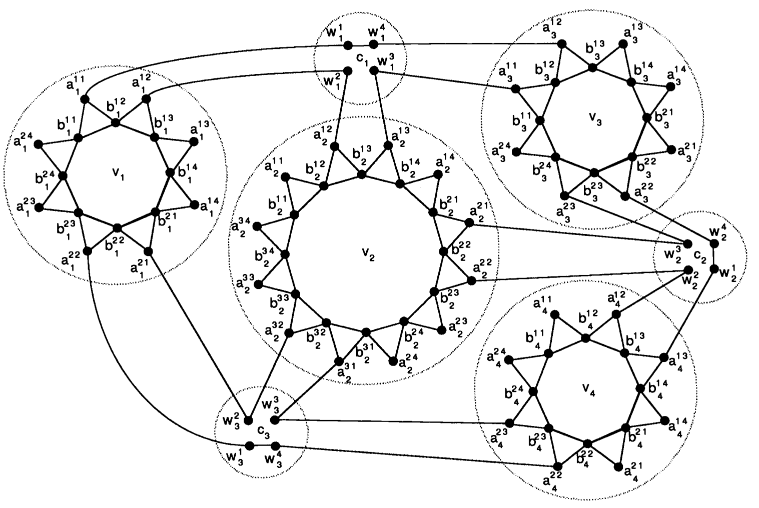


FIG. 2. *Example.* $C = \{c_1, c_2, c_3\}$; $c_1 = \{\bar{v}_1, v_2, \bar{v}_3\}$, $c_2 = \{\bar{v}_2, v_3, v_4\}$, $c_3 = \{\bar{v}_1, \bar{v}_2, v_4\}$.

sponding variable components. We call such a cycle a *clause cycle*. For example, a cycle $[w_2^1, a_4^{13}, b_4^{13}, a_4^{12}, w_2^2, a_2^{22}, b_2^{22}, a_2^{21}, w_2^3, a_3^{23}, b_3^{23}, a_3^{22}, w_2^4, w_2^1]$ is a clause cycle for clause $c_2$ in Fig. 2. Since, by assumption, each such cycle is broken, at least one vertex labeled $b$ in the three variable components must be deleted. Since the variable or its complement corresponding to such a vertex is assigned *true*, the clause corresponding to this cycle of length 13 is *true*. Therefore, there is a truth assignment for $U$ such that each clause in $C$ is true.

On the other hand, suppose that there is a truth assignment for $U$ such that each clause $c_j$ in $C$ is true. If $v_i$ is assigned *true* (respectively, *false*), then delete $n_i$ pairs of vertices $b_i^{k2}$ and $b_i^{k4}$ (respectively, $b_i^{k1}$ and $b_i^{k3}$) for $k = 1, 2, \cdots, n_i$, for each $i = 1, 2, \cdots, n$. As mentioned before, the removal of these vertices breaks all $4n_i$ triangles and leaves $2n_i$ paths of length 2 in the variable component corresponding to $v_i$. It also breaks all clause cycles. Furthermore, it is clear that all the other (possibly odd length) cycles are eliminated because each variable component is chopped into $2n_i$ paths of length 2. Therefore, the remaining graph does not contain any odd length cycles and hence it is bipartite. This completes the proof. □

**3. Via minimization.** In this section, using Theorem 2, we prove that the via minimization decision problem for *two* layers is NP-complete even if the maximum junction degree is limited to four. We start with some definitions.

A *circuit* is specified by a set of modules $M$, a set of terminals $T$, and a set of nets $N$. The *terminals* in $T$ are located on the boundary of the *modules* in $M$, and each *net* specifies which terminals are to be electrically connected. Such connections are made by patterning conductive paths on one of two *layers*. Such paths are made up of straight line segments, called *wire segments*. We assume that the terminals are available on both layers and each wire segment can be assigned to either layer. Note that the vertical projection of each wire segment is fixed in the plane but its layer assignment is not specified. A point other than a terminal at which two or more wire segments meet and are electrically connected is called a *junction*. A wire segment is said to be *incident* upon a junction at which it meets. Wire segments which are incident upon the same junction are said to be *adjacent* to each other, and the number of such segments is called the *junction degree*. If wire segments incident upon the same junction are assigned to different layers, a *via* is placed at the junction to electrically connect them. If the vertical projections of two wire segments that are not electrically connected intersect, they are said to *cross* each other. A layer assignment is said to be *valid* if no two wire segments that cross each other are assigned to the same layer and no two adjacent wire segments are assigned to different layers without a via.

The *via minimization decision problem* (VM) for two layers is defined as follows:

**Via Minimization (VM).**

**Instance:** A set $M = \{m_i \mid 1 \leqq i \leqq p\}$ of modules, a set $T = \{t_i \mid 1 \leqq i \leqq q\}$ of terminals, a set $W = \{w_i \mid 1 \leqq i \leqq r\}$ of wire segments whose vertical projections are fixed in the plane, and an integer $k \geqq 0$.

**Question:** Is there a valid layer assignment for $W$ which requires $k$ or fewer vias using two layers?

In order to show a transformation from the VDB problem for planar graphs to the VM problem, we will use the sublayout $L$ shown in Fig. 3. $L$ has a single module $m$, two terminals $t$ and $t'$ on its boundary, and wire segments $w$ and $w'$ which connect terminals $t$ and $t'$ to junctions $j$ and $j'$, respectively. Note that the terminals $t$ and $t'$ belong to different nets and thus the wires $w$ and $w'$ must be assigned to different layers.

Let $G = (V, E)$ be a planar graph. A *straight line planar embedding* of $G$ is a planar embedding of $G$ in which every edge in $G$ is represented by a straight line segment. It is known [3], [5] that such an embedding of $G$ can be obtained in polynomial time, and we will denote it by $\bar{G}$. For simplicity, we call a straight line segment in $\bar{G}$ which represents an edge $(x, y)$ in $G$, a *segment* $(x, y)$, and the endpoints of the line segment which represent vertices $x$ and $y$, *points* $x$ and $y$, respectively.

We first create a small region surrounding each segment $(x, y)$ in $\bar{G}$ so that no two such regions overlap. We then replace each segment $(x, y)$ by a sublayout $L$ in such a manner that $L$ is completely within the region surrounding $(x, y)$ and junctions $j$ and $j'$ coincide with points $x$ and $y$, respectively.

We denote the resultant layout by $L(G)$. Figure 4 shows an example graph $G_1$ and its corresponding layout $L(G_1)$. Note that for each cycle in $G$, there is a cycle of sublayouts or modules in $L(G)$. We call such a cycle of modules an *m-cycle*. If the number of modules in an $m$-cycle is odd (respectively, even), it is called an *odd* (respectively, *even*) $m$-cycle. Consider the $m$-cycle consisting of modules $m_1$, $m_2$, and $m_3$ shown in Fig. 4(b). Suppose that wire segment $w_1$ is assigned to layer 1. Then $w'_1$ must be assigned to layer 2. To avoid a via at junction $j_2$, $w_2$ should be assigned to layer 2. Consequently, $w'_2$ must be assigned to layer 1 and hence $w'_3$ should be assigned to layer 1. This implies that $w_3$ must be assigned to layer 2. In order to electrically connect the wire segments $w_3$ and $w_1$, a via needs to be placed at junction $j_1$. It is not difficult to see that a via is always required to have a valid layer assignment if there is an odd $m$-cycle in $L(G)$.

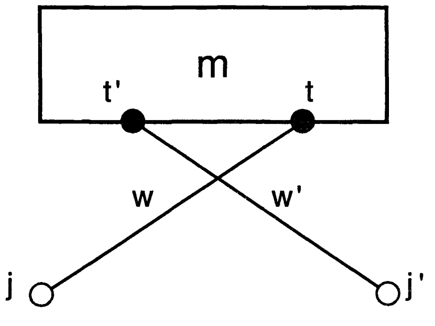


FIG. 3. *Sublayout L.*

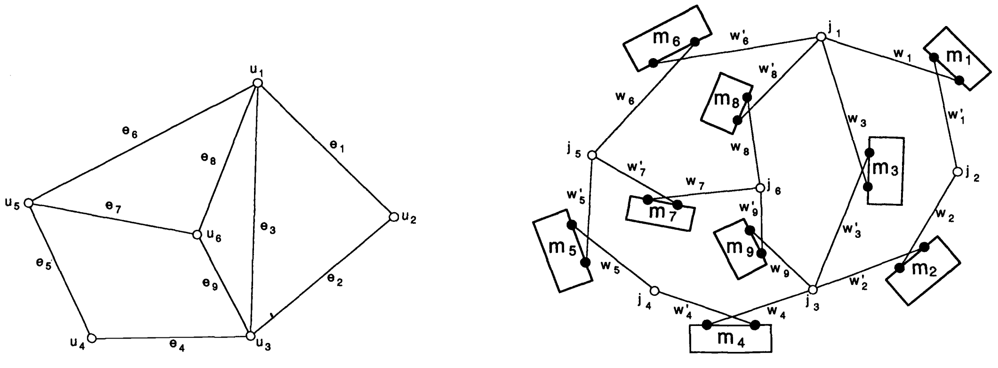


FIG. 4. *An example graph* $G_1$ *and its corresponding layout* $L(G_1)$. (a) $G_1$. (b) $L(G_1)$.

On the other hand, suppose that there is no odd $m$-cycle in $L(G)$. Then a valid layer assignment for $L(G)$ which requires no via will be obtained in the following manner. Assign an aribitrary wire segment and all of its adjacent wire segments to layer 1. Find wire segments that cross the wire segments just assigned to layer 1. Assign those segments to layer 2. Then find unassigned wire segments that cross the segments just assigned to layer 2 and assign them to layer 1. Repeat this process until all segments are assigned to one of the layers. Since there is no odd $m$-cycle, no conflict on layer assignment would occur. Therefore, we have the following lemma.

LEMMA 1. *There exists a valid layer assignment for layout* $L(G)$ *which requires no via if and only if it is free of odd m-cycles.*

We are now ready to show the NP-completeness of the VM problem even when the maximum junction degree is limited to four.

THEOREM 3. *The* VM *problem is* NP-*complete even when the maximum junction degree is limited to four.*

*Proof.* It is easy to see that the VM problem belongs to the class NP. Thus, it suffices to show that the VDB problem for a planar graph $G$ with $\Delta(G) = 4$ is transformable in polynomial time to the VM problem.

Let $G = (V, E)$ be a planar graph such that $\Delta(G) = 4$ and $k$ be a nonnegative integer. We construct a layout $L(G)$ in the manner described above.

Suppose that there is a set of vertices $V' \subset V$ such that $|V'| \leqq k$ and $G^v(V - V')$ is bipartite. Let $J$ be a set of junctions in $L(G)$ which correspond to the vertices in $V'$. We first delete from $L(G)$ all junctions in $J$ and the wire segments incident upon them. Let $L^d(G)$ denote the resultant layout. Figure 5(a) depicts such a layout which is obtained by deleting junction $j_1$ from $L(G_1)$ of Fig. 4(b). Note that there are two types of "*degenerated*" sublayouts in $L^d(G)$. Sublayouts of Type 1 (respectively, 0) are those that consist of a module and one (respectively, no) wire segment. In Fig. 5(a), sublayouts containing modules $m_1$, $m_3$, $m_6$, and $m_8$ are of Type 1. If junction $j_2$ were also deleted, the sublayout containing module $m_1$ would be of Type 0. We then delete those degenerated sublayouts from $L^d(G)$. Let $L'(G)$ be the resultant layout. Since $L'(G)$ does not contain any odd $m$-cycle, by Lemma 1, there exists a valid layer assignment for $L'(G)$ which requires no via. Figure 5(b) illustrates such a layer assignment for $L'(G_1)$. Then, based on this layer assignment, we can obtain a valid layer assignment for $L^d(G)$ as follows: For each degenerated sublayout of Type 1, assign its only wire segment to the same layer as all of its adjacent wire segments in $L'(G)$. Such a layer assignment for $L'(G_1)$ is shown in Fig. 5(c).

We now find layer assignments for those wire segments which are incident upon the junctions in $J$. For each wire segment that is missing from some degenerated sublayout

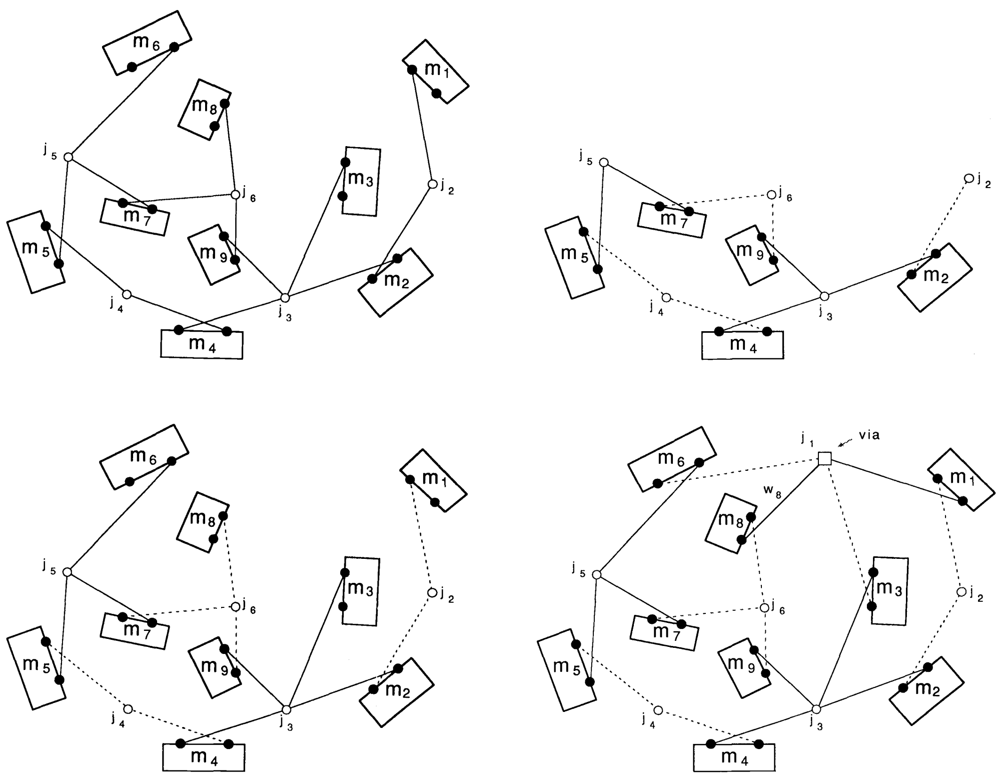


FIG. 5. *Illustrations for the proof of Theorem* 3. (a) $L(G_1^d)$. (b) *A valid layer assignment for* $L(G_1')$. (c) *A valid layer assignment for* $L(G_1^d)$. (d) *A valid layer assignment for* $L(G_1)$.

of Type 1, if the remaining wire segment is assigned to layer 1, we assign the missing segment to layer 2 and vice versa. We assign a pair of wire segments in each degenerated sublayout of Type 0 to different layers arbitrarily. Because vias are placed at those junctions in $J$, necessary electrical connections will be made. Therefore, a valid layer assignment exists for $L(G)$ which requires $k$ or fewer vias, since $|J| \leqq k$. A final valid layer assignment for $L(G_1)$ is shown in Fig. 5(d).

Conversely, suppose that there is a set of $k$ or fewer vias for which a valid layer assignment for $L(G)$ exists. Let $J$ be the set of junctions at which the vias are placed. Let $V'$ be the set of vertices that corresponds to the junctions in $J$.

Consider the vertex-deleted subgraph $G^v(V - V')$ and its corresponding layout $L(G^v(V - V'))$. It is not difficult to see that $L(G^v(V - V'))$ is obtained by deleting all such sublayouts that contain at least one wire segment that is incident upon some junction in $J$. Since the layer assignment for $L(G^v(V - V'))$ is valid and requires no via, by Lemma 1 there is no odd $m$-cycle in $L(G^v(V - V'))$. This implies that there is no odd length cycle in $G^v(V - V')$ and hance $G^v(V - V')$ is bipartite. Therefore, there exists a set of vertices $V'$ such that $|V'| = |J| \leqq k$ and $G^v(V - V')$ is bipartite.

We have proved that there is a set of vertices $V' \subset V$ such that $|V'| \leqq k$ and $G^v(V - V')$ is bipartite if and only if a valid layer assignment exists for $L(G)$ which requires $k$ or fewer vias. Since the construction of $L(G)$ only requires replacing each segment in $\bar{G}$ with a sublayout $L$, it can easily be accomplished in polynomial time. Furthermore, the number of wire segments incident upon a junction in $L(G)$ is the same as the number of segments incident upon the corresponding point in $\bar{G}$. Since $\Delta(G) =$

4, the maximum junction degree in $L(G)$ is limited to four. This completes the proof of the theorem. □

*Remark* 2. Using the same arguments as in Naclerio, Masuda, and Nakajima [15], we can establish the NP-completeness of the VM problem for two layers under any combination of the following two constraints as long as the maximum junction degree is limited to four or more:

(1) The layout is restricted to be grid-based.

(2) Vias are placed only at the junctions that existed in the input layout.

**4. Conclusion.** We have presented complexity results for the vertex-deletion graph bipartization (VDB) problem. These results completely close the gap between the polynomially solvable cases and NP-complete cases for general and planar graphs as shown in Table 1. We have also shown that the via minimization decision problem is NP-complete for two layers when the maximum junction degree is limited to four. Quite recently, Molitor [13] showed that the problem of assigning wire segments to three or more layers without using vias is NP-complete when the maximum junction degree is limited to four. Since the via minimization problem for two layers is solvable in polynomial time when the maximum junction degree is limited to three [2], [13], [14], [16], our result has completely settled the complexity issues on via minimization.

**Acknowledgment.** The authors would like to thank the referee for providing a simple proof of Theorem 1.

## REFERENCES

[1] K. AOSHIMA AND M. IRI, *Comments on F. Hadlock's paper: Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput., 6 (1977), pp. 86–87.

[2] R-W. CHEN, Y. KAJITANI, AND S-P. CHAN, *A graph-theoretic via minimization algorithm for two-layer printed circuit boards*, IEEE Trans. Circuits Systems, CAS-30 (1983), pp. 284–299.

[3] N. CHIBA, T. NISHIZEKI, S. ABE, AND T. OZAWA, *A linear time algorithm for embedding planar graphs using PQ-trees*, J. Comput. System Sci., 30 (1985), pp. 54–76.

[4] S. EVEN, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[5] I. FARY, *On straight line representation of planar graphs*, Acta Sci. Math. (Szeged), 11 (1948), pp. 229–233.

[6] M. R. GAREY AND D. S. JOHNSON, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 826–834.

[7] M. R. GAREY, D. S. JOHNSON, AND L. STOCKMEYER, *Some simplified NP-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237–267.

[8] F. HADLOCK, *Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput., 4 (1975), pp. 221–225.

[9] A. HASHIMOTO AND J. STEVENS, *Wire Routing by Optimizing Channel Assignment within Large Apertures*, Proc. 8th Design Automation Workshop, Atlantic City, NJ, June 1971, pp. 155–169.

[10] Y. KAJITANI, *On Via Hole Minimization of Routing on a 2-Layer Board*, Proc. 1980 IEEE Internat. Conference on Circuits and Computers, New York, NY, October 1980, pp. 295–298.

[11] M. S. KRISHNAMOORTHY AND N. DEO, *Node-deletion NP-complete problems*, SIAM J. Comput., 8 (1979), pp. 619–625.

[12] D. LICHTENSTEIN, *Planar formulae and their uses*, SIAM J. Comput., 11 (1982), pp. 329–343.

[13] P. MOLITOR, *On the Contact-Minimization Problem*, Proc. 4th Annual Symp. on Theoretical Aspects of Computer Science, Passau, West Germany, February 1987; Lecture Notes in Computer Science 247, Springer-Verlag, New York, Berlin, 1987, pp. 420–431.

[14] N. J. NACLERIO, S. MASUDA, AND K. NAKAJIMA, *Via Minimization for Gridless Layouts*, Proc. 24th ACM/IEEE Design Automation Conference, Miami Beach, FL, June 1987, pp. 159–165.

[15] ———, *The via minimization problem is NP-complete*, IEEE Trans. Comput., to appear.

[16] R. P. PINTER, *Optimal Layer Assignment for Interconnect*, Proc. Internat. Conference on Circuits and Computers, New York, NY, September 1982, pp. 398–401.

[17] M. YANNAKAKIS, *Node- and Edge-deletion NP-complete Problems*, Proc. 10th Annual ACM Symp. on Theory of Computing, San Diego, CA, October 1978, pp. 253–264.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 48–50, February 1989

# IMPOSSIBILITY OF DECOMPOSING THE COMPLETE GRAPH ON $n$ POINTS INTO $n - 1$ ISOMORPHIC COMPLETE BIPARTITE GRAPHS*

D. DE CAEN† AND D. G. HOFFMAN‡

**Abstract.** It is proved that $K_{2rs}$ cannot be edge-partitioned into copies of $K_{r,s}$ if $r$ and $s$ are greater than one. This answers a question raised in a recent paper of Granville, Moisiadis, and Rees. [*Congr. Numer.*, 61 (1988), pp. 241–248].

A theorem of Graham and Pollak [3, p. 105] asserts that the complete graph $K_n$ can be edge-partitioned into no fewer than $n - 1$ complete bipartite graphs (bicliques). It is easy to see that there are many ways to decompose $K_n$ into $n - 1$ bicliques. Zaks [9] discusses a connection between inequivalent minimal biclique partitions and inequivalent arrangements of neighborly cubes in Euclidean space. Recently, Granville, Moisiadis, and Rees [4] considered the problem of decomposing into $n - 1$ isomorphic bicliques, say copies of $K_{r,s}$ for given integers $r$ and $s$. It is clear that $n = 2rs$ in this case.

By examining the edges incident to any point of $K_n$, it follows that $r$ and $s$ must be relatively prime. Hence we may take $1 \leqq r < s$ with greatest common divisor (GCD) equal to one. In [4], it is shown that $K_{2s}$ can be decomposed into copies of $K_{1,s}$, but $K_{4s}$ cannot be decomposed into copies of $K_{2,s}$. We complete the solution of this problem by proving the following result.

THEOREM. *Let $2 \leqq r < s$. Then $K_{2rs}$ cannot be edge-partitioned into copies of $K_{r,s}$.*

We begin by giving the main idea of the proof, before getting to details. Let $n = 2rs$ and suppose that $K_n$ is partitioned into $n - 1$ bicliques $B_i = (R_i, S_i)$, where $|R_i| = r$ and $|S_i| = s$ for $i = 1$ to $n - 1$. Orient the edges of $B_i$ from $R_i$ to $S_i$. This gives us an associated tournament $T$ on $n$ vertices. The adjacency matrix $M(T)$ of $T$ satisfies

$$(1) \qquad M(T) = M(B_1) + \cdots + M(B_{n-1})$$

where each $M(B_i)$ is a "rectangle," i.e., its 1's form an $r \times s$ submatrix. Thus $M(B_i)$ has rank one, and so by (1) $M(T)$ has rank at most $n - 1$; in particular $M(T)$ is singular. On the other hand, various counting arguments will tell us a lot about the structure of $M(T)$: it is a subdirect sum of two nontrivial regular tournament matrices. It is a simple matter to show that such a matrix is nonsingular, a contradiction that will complete the proof of our theorem.

We recall some calculations made in [4]. In what follows $r$ and $s$ are given integers with $1 \leqq r < s$ and GCD $(r, s) = 1$. We fix a decomposition of $K_{2rs}$ into copies of $K_{r,s}$, say $B_i = (R_i, S_i)$ for $i = 1$ to $2rs - 1$. Given a point $x$, let $d_r(x)$ be the number of sets $R_i$ incident to $x$ and $d_s(x)$ the number of sets $S_i$ incident to $x$. Clearly,

$$(2) \qquad \text{for all } x, \quad s d_r(x) + r d_s(x) = 2rs - 1.$$

Note that $d_s(x)$ is not zero, otherwise we see by (2) that $s$ divides $(-1)$, contradicting $s > 1$. Similarly, $d_r(x) > 0$ as long as $r > 1$.

Since $r$ and $s$ are relatively prime, there is a unique $m \in \{0, 1, 2, \cdots, r - 1\}$ such that $ms$ is congruent to $(-1)$ modulo $r$. Denote this $m$ by $\alpha(r, s)$. It follows easily from

* Received by the editors April 27, 1988; accepted for publication August 4, 1988.

† Department of Mathematics and Statistics, Queen's University, Kingston, Ontario, Canada K7L 3N6. This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant URFIN 142.

‡ 120 Mathematics Annex, Auburn University, Auburn, Alabama 36849. This research was supported by the National Security Agency under grant 4-21100 MDA-904-88-H-2005.

(2) that for each $x$, $d_r(x) = \alpha(r, s)$ or $\alpha(r, s) + r$. Symmetrically, $d_s(x) = \alpha(s, r)$ (whenever $d_r(x) = \alpha(r, s) + r$) or $d_s(x) = \alpha(s, r) + s$ (whenever $d_r(x) = \alpha(r, s)$). Let $A$ be the set of points $x$ such that $d_r(x) = \alpha(r, s)$ and $B$ the remaining set of points. We have

(3) $$|A| + |B| = 2rs,$$

(4) $$|A|\alpha(r,s) + |B|(\alpha(r,s)+r) = r(2rs-1),$$

(5) $$|A|(\alpha(s,r)+s) + |B|\alpha(s,r) = s(2rs-1).$$

From these equations, we find that

(6) $$|A| = 2s\alpha(r,s)+1,$$

(7) $$|B| = 2r\alpha(s,r)+1.$$

We remark that if $r = 1$, then $\alpha(1, s) = 0$ and $\alpha(s, 1) = s - 1$, so $|A| = 1$ and $|B| = 2rs - 1$. If $r \geqq 2$, then both $\alpha(r, s)$ and $\alpha(s, r)$ are nonzero, and so $|A|$ and $|B|$ are odd integers greater than one.

We now define a tournament $T$ as follows: orient the $rs$ edges of each biclique $B_i = (R_i, S_i)$ from $R_i$ to $S_i$. let $p^+(x)$ denote the out-degree of $x$ in $T$, i.e., the number of arcs of the form $x \twoheadrightarrow y$. Let $p^-(x)$ denote the in-degree of $x$. It follows easily from (6), (7), and the definition of $T$ that

(i) If $x \in A$, then $p^+(x) = s\alpha(r, s)$ and $p^-(x) = r\alpha(s, r) + rs$;

(ii) If $x \in B$, then $p^+(x) = r\alpha(r, s) + rs$ and $p^-(x) = r\alpha(s, r)$.

We claim that all arcs of $T$ between $A$ and $B$ are directed from $B$ to $A$. To see this, note that

(8) $$\sum_{x\in A} p^+(x) \geqq \binom{|A|}{2}$$

because each pair in $A$ is oriented one way, and so the outgoing arcs from the points in $A$ will cover all these pairs. Furthermore, this observation tells us that if there is equality in (8), then all arcs leaving a point in $A$ go to a point in $A$, so all arcs between $A$ and $B$ go from $B$ to $A$. Using(6) and (i) above, we see that equality does indeed hold in (8), proving our claim.

Let $T_A$ be the subtournament of $T$ induced on the point-set $A$. By the preceding remarks, we find that for each $x \in A$, the in-degree of $x$ within $T_A$ is $p^-(x) - |B| = r\alpha(s, r) + rs - 2r\alpha(s, r) - 1 = s\alpha(r, s)$, which equals the out-degree of $x$ within $T_A$. Thus $T_A$ is a regular tournament on $2s\alpha(r, s) + 1$ points. Similarly, $T_B$ is regular. In summary, the adjacency matrix of $T$ has the following form:

$$M(T) = \left[\begin{array}{c|c} M(T_A) & 0 \\ \hline J & M(T_B) \end{array}\right]$$

where $J$ is the $|B| \times |A|$ all-ones matrix. Also, if $r > 1$, then, as noted earlier, $|A|$ and $|B|$ are odd integers greater then one. It is an easy matter to show that a regular tournament on more than one point has a nonsingular adjacency matrix. (This was proved by Brauer and Gentry [1]; another very short proof is given in [2].) Thus $M(T_A)$ and $M(T_B)$ are nonsingular, and hence so is $M(T)$.

As noted in the introduction, we have the contradiction that $M(T)$ is also singular, since it is the sum of $n - 1$ rank-one matrices $M(B_i)$. This completes the proof of our theorem. We add that in the case $r = 1$, the argument based on inequality (8) leads to a complete determination of all decompositions of $K_{2s}$ into copies of $K_{1,s}$. Indeed, by the argument of (8) the associated tournament has a sink (the singleton set $A$); deleting the sink leaves us with a regular tournament on $2s - 1$ points. Conversely, given such a tournament $T$, we can obtain a partition of $K_{2s}$ into $K_{1,s}$'s as follows: add an extra point $\infty$; for each point $x$ of $T$ take the $(s - 1)$ arcs leaving $x$ and add the edge $x\infty$ to make a $K_{1,s}$; this evidently gives the required decomposition. Thus there is a bijective correspondence between regular tournaments on $2s - 1$ points and edge-partitions of $K_{2s}$ into copies of $K_{1,s}$. Of course, it remains to "determine" or enumerate all regular tournaments, a very difficult problem.

See Huang and Rosa [5] for a general study of edge-partitions of the complete graph into isomorphic bicliques. In particular, Rosa [7] showed that $K_n$ can always be decomposed into $n$ copies of $K_{r,s}$ (where now, of course, $rs = \frac{1}{2}(n - 1)$). Thus our nonexistence theorem only applies to the extremal decompositions into $(n - 1)$ bicliques.

We conclude by mentioning that an arbitrary $n \times n$ tournament matrix $M$ has rank at least $n - 1$, over a field of characteristic zero. This is a slightly stronger statement than the Graham–Pollak theorem. Indeed, suppose that $K_n$ is decomposed into $k$ bicliques $B_i = (X_i, Y_i)$. Form a tournament $T$ by orienting each $B_i$ from $X_i$ to $Y_i$. Then $M(T) = M(B_1) + \cdots + M(B_k)$ where each $B_i$ has rank one, so $M(T)$ has rank at most $k$. Since the tournament matrix $M(T)$ has rank at least $n - 1$, then $k \geqq n - 1$, as desired. Proofs that $M$ has rank at least $n - 1$ can be patterned after the known proofs of the Graham–Pollak theorem. For example, here is Gregory's proof, inspired by Peck's proof [6]. By definition, the tournament matrix $M$ satisfies $M + M^t = J - I$, where $M^t$ is the transpose of $M$, $J$ is the $n \times n$ all-ones matrix, and $I$ the identity matrix. Note that $M - M^t$ is skew-symmetric, so all its eigenvalues lie on the imaginary axis. Hence $A = I + M - M^t$ is nonsingular. Also, $J = I + M + M^t$ has rank one. Hence $A - J = -2M^t$ has rank at least $n - 1$. (In general, we have the easy inequality rank $(X + Y) \leqq$ rank $(X)$ + rank $(Y)$. Hence rank $(A - J) \geqq$ rank $(A)$ − rank $(J)$ = $n - 1$.) Thus $M$ has rank at least $n - 1$, as desired. Another proof (in the spirit of Tverberg's proof [8]) is given in [2], which also discusses the ranks of tournament matrices over modular fields.

## REFERENCES

[1] A. Brauer and I. C. Gentry, *On the characteristic roots of tournament matrices*, Bull. Amer. Math. Soc., 74 (1968), pp. 1133–1135.

[2] D. de Caen, *The rank of tournament matrices over arbitrary fields*, preprint.

[3] R. L. Graham and H. O. Pollak, *On embedding graphs in squashed cubes*, Springer Lecture Notes 303, Springer-Verlag, New York, Berlin, 1973, pp. 99–110.

[4] A. Granville, A. Moisiadis, and R. Rees, *Bipartite planes*, Congr. Numer., 61 (1988), pp. 241–248.

[5] C. Huang and A. Rosa, *On the existence of balanced bipartite designs*, Utilitas Math., 4 (1973), pp. 55–75.

[6] G. W. Peck, *A new proof of a theorem of Graham and Pollak*, Discrete Math., 49 (1984), pp. 327–328.

[7] A. Rosa, *On certain valuations of the vertices of a graph*, in Theory of Graphs, Rome, 1966, pp. 349–355.

[8] H. Tverberg, *On the decomposition of $K_n$ into complete bipartite graphs*, J. Graph Theory, 6 (1982), pp. 493–494.

[9] J. Zaks, *How does the complete graph split into complete bipartite graphs and how are neighborly cubes arranged?*, Amer. Math. Monthly, 92 (1985), pp. 568–570.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 51–63, February 1989

# LOW RANK MATRICES WITH A GIVEN SIGN PATTERN*

P. DELSARTE AND Y. KAMP†

**Abstract.** Given an $m \times n$ sign matrix $S$, an $m \times n$ real matrix $A$ is said to be a realization of $S$ if the sign of the $(i, j)$-entry of $A$ equals the $(i, j)$-entry of $S$. This paper deals with the problem of finding low rank realization matrices $A$. It is motivated by a minimization problem in multilayer perceptrons. The subject is approached by means of the Farkas lemma, which allows characterization of the sign matrices realizable with a given rank. Based on this result and on some other standard techniques of matrix algebra such as the cyclic Fourier transform, low rank realizations are obtained for sign matrices having certain nice combinatorial structures. Furthermore, the paper includes an elementary lower bound on the rank and a counting of realizable sign vectors.

**Key words.** sign matrices, low rank matrices, neural networks, multilayer perceptrons, Farkas lemma

**AMS(MOS) subject classifications.** 15A39, 15A03, 52A25

## 1. Introduction.

This paper is devoted to the general question of constructing real $m \times n$ matrices of low rank under the constraint that each entry is nonzero and has a given sign. The problem arises from an interesting topic in neural networks or, more specifically, in multilayer perceptrons [4], [6]. In this application, the rank of a realization matrix can be interpreted as the number of elements in a hidden layer, which motivates a search for low rank solutions.

A realization matrix $A$ of rank $k$ can be factorized in the form $A = X^T Y$ for suitable $k \times m$ and $k \times n$ matrices $X$ and $Y$. The sign $m$-vectors realizable by means of a given $k \times m$ matrix $X$ (which are the sign vectors of the columns of all possible matrices $A = X^T Y$) can be characterized in explicit terms with the help of Gordan's transposition theorem [8]. This result, which belongs to the classical duality principle of linear programming, is one of the main tools used in the paper. It is given in § 2, after some preliminary definitions, further details concerning the motivation, and a simple lower bound for the rank of a realization.

It proves quite interesting to consider the special case where all elements of the first row of $X$ are positive or, equivalently, are equal to unity. In the neural network application, this corresponds to the "bias assumption" [6]. In this case, the sign vectors realizable by means of $X$ can be characterized, in simple geometric terms, in the framework of a $(k - 1)$-dimensional affine space. The result is given in § 3, together with some illustrative examples for small values of $k$. In particular, an explicit rank 3 realization is indicated for the square matrix having plus signs on the main diagonal and minus signs elsewhere, which is especially significant in the neural network application.

Section 4 contains a detailed investigation of certain structured sign matrices for which low rank realizations can be obtained by simple algebraic methods. It is first shown that a circulant sign matrix admits a circulant realization whose rank does not exceed one plus the number of sign discontinuities. Next, the question of the "direct sum" of sign matrices is examined. The general result obtained for this question implies, in particular, that any sign matrix having at most two plus signs in each row and each column admits a realization of rank 4.

Finally, § 5 is devoted to the problem of determining the number of distinct sign vectors realizable by means of a given $k \times m$ matrix $X$. This number is shown to be a

* Received by the editors December 21, 1987; accepted for publication (in revised form) May 23, 1988.

† Philips Research Laboratory Brussels, Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium.

constant, depending only on $k$ and $m$, for the class of matrices $X$, all $k \times k$ submatrices of which are nonsingular. A closed form expression is derived for that constant. It coincides with the Cameron–Winder upper bound on the number of threshold functions of $k-1$ variables defined on $m$ points [1], which corresponds to a different setting of the same combinatorial problem.

**2. Definitions, motivations, and preliminary results.** For positive integers $m$ and $n$, consider an $m \times n$ matrix $S = [s_{i,j}\colon 1 \leqq i \leqq m,\ 1 \leqq j \leqq n]$ consisting of *signs*, i.e., elements $s_{i,j}$ of the two-set $\{+, -\}$. In the sequel, the signs $+$ and $-$ are sometimes interpreted as the real numbers 1 and $-1$. An $m \times n$ matrix $A = [a_{i,j}\colon 1 \leqq i \leqq m,\ 1 \leqq j \leqq n]$ of nonzero real numbers $a_{i,j}$ is said to *realize* $S$ if it satisfies

$$\operatorname{sgn}(A) = S, \tag{2.1}$$

in the sense that the $(i, j)$-entry $a_{i,j}$ of $A$ has sign $s_{i,j}$ for all $i$ and $j$. This paper is concerned with the problem of finding *low rank* matrices $A$ that realize a given sign matrix $S$.

Let us make a first elementary observation. Assume that, for each sign vector $\mathbf{s} \in \{+, -\}^m$, either $\mathbf{s}$ or $-\mathbf{s}$ (or both) occurs as a column of $S$. (Of course, this implies $n \geqq 2^{m-1}$.) Then any matrix $A$ realizing $S$ has full row rank (i.e., has rank $m$). To see this, suppose there exists a nonzero real $m$-tuple $(\lambda_1, \cdots, \lambda_m)$ satisfying $\lambda_1 a_{1,j} + \cdots + \lambda_m a_{m,j} = 0$ for $1 \leqq j \leqq n$. This is impossible since, for a certain value of $j$ (depending on the signs of the coefficients $\lambda_i$), all nonzero numbers $\lambda_i a_{i,j}$ have the same sign. Conversely, if, for some $\mathbf{s}$, neither $\mathbf{s}$ nor $-\mathbf{s}$ occurs as a column of $S$, then there exists a realization matrix $A$ of rank less than $m$. Without loss of generality, consider the case $\mathbf{s} = (+, \cdots, +)^T$. It is easily seen that some real values can be assigned to the elements $a_{i,j}$, under the constraint (2.1), in such a way that the sum of the rows of $A$ vanishes, which proves the claim. (Note that this property cannot be used recursively, in general, because the required value assignments can become inconsistent.)

The former result leads immediately to a lower bound for the rank of a realization matrix. Indeed, if $S$ contains an $r \times n$ submatrix $S'$ whose column set intersects each two-set $\{\mathbf{s}', -\mathbf{s}'\}$, with $\mathbf{s}' \in \{+, -\}^r$, then the rank of any realization $A$ of $S$ is bounded from below by

$$\operatorname{rank}(A) \geqq r. \tag{2.2}$$

Of course, there is a similar result for the transpose matrix. The lower bound on rank $(A)$ resulting from this argument is very small with respect to the sizes $m$ and $n$; it is at most logarithmic in min $(m, n)$. In general, this bound is not achievable.

Assume $A$ to have rank $k$, for a certain integer $k$ with $1 \leqq k \leqq \min(m, n)$. Then it can be factorized in the following form:

$$A = X^T Y, \tag{2.3}$$

where $X$ is a $k \times m$ matrix and $Y$ is a $k \times n$ matrix, both of rank $k$. Denote by $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ and by $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$ the vectors in $\mathbf{R}^k$, which are the successive columns of the matrices $X$ and $Y$. The constraint (2.1) can be written as

$$\operatorname{sgn}(\mathbf{x}_i^T \mathbf{y}_j) = s_{i,j} \quad \text{for all } i \text{ and } j. \tag{2.4}$$

Let us now provide a motivation for that problem, leading to further specific questions. Neural networks is a generic name for highly distributed architectures of computing elements that can be used for a wide class of applications, among which classification and associative memory are especially important [3], [4], [7]. *Multilayer perceptrons* form a subclass of neural networks in which the nonlinear computing elements, called *units*, are arranged in a feed-forward structure of successive layers [4], [6]. The values of the units in the input layer ($l = 0$) are externally dictated. For each of the following layers ($l = 1, 2, \cdots, L$), each unit computes a weighted sum of the unit values in the preceding layer and passes this result through an approximation $\sigma_l$ of the sign function. Formally, we can write

$$\mathbf{z}_l = \sigma_l(W_l \mathbf{z}_{l-1}), \qquad l = 1, 2, \cdots, L, \tag{2.5}$$

where $\mathbf{z}_l$ is the vector formed by the values of the $m_l$ units in layer $l$ and $W_l$ is a suitable $m_l \times m_{l-1}$ real matrix. Here $\sigma_l$ denotes a well-defined nonlinear function; it is applied componentwise to the vector $W_l \mathbf{z}_{l-1}$.

When the multilayer perceptron is used as a classifier, the nonlinear function $\sigma_L$ at the output layer is the sign function itself. For this operation mode, a real input vector $\mathbf{z}_0$ propagates through the structure according to (2.5) and produces an output vector $\mathbf{z}_L$, which is the binary code assigned to $\mathbf{z}_0$. Now considering a collection of $n$ input vectors, let us denote by $Z_l$ the $m_l \times n$ matrix whose columns are the unit values $\mathbf{z}_l$ in layer $l$ for each of the $n$ input patterns $\mathbf{z}_0$. In particular, the set of equations for the output layer reads

$$Z_L = \operatorname{sgn}(W_L Z_{L-1}). \tag{2.6}$$

The input matrix $Z_0$ is determined by the set of vectors to be classified. Usually, the output matrix $Z_L$ results from the particular classification problem at hand.

It is customary to introduce a *bias* that corresponds to adding a constant offset to the weighted sum computed by each unit. This is reflected in (2.6) by increasing the respective column and row sizes of $W_L$ and $Z_{L-1}$, and by imposing that all elements in a column of $W_L$ (or in a row of $Z_{L-1}$) be equal to one.

In many practical applications, the number of intermediate layers ($0 < l < L$), also called *hidden layers* in the neural network literature, can in principle be reduced to one or two [4]. The number of units to be put in the hidden layers, however, is not rigidly fixed by the specifications of the classification problem. It is therefore meaningful to examine what the theoretical minimum number of units is, especially in the last hidden layer ($l = L - 1$, immediately before the output). This corresponds to the algebraic question mentioned above, with $S = Z_L, X = W_L^T, Y = Z_{L-1}, m = m_L$, and $k = m_{L-1}$. Thus it is seen that part of the problem of the minimum number of hidden units can be viewed as a problem of minimum rank realization for a prescribed sign matrix.

An important example is the case where the classes are in one-to-one correspondence with the output units. For each input vector $\mathbf{z}_0$ belonging to class $i$, the corresponding output vector $\mathbf{z}_L$ should have a plus sign in position $i$ and a minus sign elsewhere. The basic matrix $Z_L$ for such a situation has the form

$$Z_L = \begin{bmatrix} + & - & \cdots & - \\ - & + & \cdots & - \\ \vdots & \vdots & & \vdots \\ - & - & \cdots & + \end{bmatrix}. \tag{2.7}$$

In fact, any matrix $Z_L$ occurring in that example can be obtained from (2.7) by permutation and repetition of the columns. This concludes our incursion into neural networks.

Going back to the initial algebraic setting (2.3), (2.4), we now examine the question from a different point of view. Let there be given a $k \times m$ matrix

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m] \quad \text{with } \mathbf{x}_i \in \mathbf{R}^k. \tag{2.8}$$

We assume rank $(X) = k$. In addition, without real loss of generality, we assume that the columns $\mathbf{x}_i$ of $X$ are distinct. We are interested in characterizing *the set $\mathscr{S}(X)$ of sign vectors $\mathbf{s} \in \{+, -\}^m$ that are realizable by means of $X$*. The formal definition is

$$(2.9) \qquad \mathscr{S}(X) = \{\operatorname{sgn}(X^T\mathbf{y}) : \mathbf{y} \in \mathbf{R}^k, \mathbf{x}_i^T\mathbf{y} \neq 0 \text{ for } 1 \leqq i \leqq m\}.$$

This is clearly relevant to the initial problem. Indeed, there exists a realization $A = X^TY$ of the sign matrix $S$ (for the given $X$) if and only if all columns of $S$ belong to the set $\mathscr{S}(X)$. Note that the computation of a $k \times n$ matrix $Y$ associated with $X$ resorts to classical linear programming.

By using a modified version of the Farkas duality lemma, known as the *Gordan transposition theorem* (see [8, p. 95]), we immediately obtain the following characterization.

THEOREM 1. *The sign vector $\mathbf{s} = (s_1, s_2, \cdots, s_m)^T$, with $s_i \in \{+, -\}$, is realizable by means of the $k \times m$ matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$ if and only if the convex hull of the points $s_1\mathbf{x}_1, s_2\mathbf{x}_2, \cdots, s_m\mathbf{x}_m$ (in the Euclidean space $\mathbf{R}^k$) does not contain the origin $\mathbf{0}$.*

*Proof.* The property $\mathbf{s} \in \mathscr{S}(X)$ can be written as $s_i\mathbf{x}_i^T\mathbf{y} > 0$, for $i = 1, 2, \cdots, m$, for a suitable $\mathbf{y} \in \mathbf{R}^k$. By Gordan's theorem, there exists such a vector $\mathbf{y}$ if and only if there exists no nonzero $m$-tuple $(\lambda_1, \cdots, \lambda_m)$ of nonnegative real numbers $\lambda_i$ satisfying $\sum \lambda_i s_i \mathbf{x}_i = 0$. This result is exactly equivalent to the statement of the theorem. □

*Remarks.* (i) When $S$ admits a realization of rank $k$, then it clearly admits a realization of rank $k'$ for all $k'$ with $k \leqq k' \leqq \min(m, n)$. Therefore, realizable with rank $k$ has the same meaning as realizable with rank not exceeding $k$.

(ii) The set $\mathscr{S}(X)$ remains exactly the same when the matrix $X$ is replaced by $RXD$ where $R$ is any $k \times k$ nonsingular matrix and $D$ is any $m \times m$ diagonal matrix with positive diagonal entries.

(iii) It is sometimes useful to consider the case where all $k \times k$ submatrices of $X$ are nonsingular. This will be referred to as the *nondegenerate case*.

**3. Affine type realizations.** Let us now put a restrictive condition on the matrix $X$; it is not only interesting from a theoretical viewpoint but also relevant from the application viewpoint. It is required that the sign $m$-vector $\mathbf{s}_0 = (+, +, \cdots, +)^T$ be realizable by means of $X$. In other words, all points $\mathbf{x}_i$ have to belong to the same half-space $\mathbf{x}^T\mathbf{y}_0 > 0$ for a certain $\mathbf{y}_0 \in \mathbf{R}^k$. In view of Theorem 1, this means exactly that the convex hull of $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ does not contain the origin. Without loss of generality we can assume, in this case, that *all points $\mathbf{x}_i$ belong to the same affine hyperplane of the space $\mathbf{R}^k$*; see Remark (ii) in § 2.

As a consequence, the geometric dimension of the problem is reduced from $k$ to $k - 1$. More specifically, we can assume that the first component of $\mathbf{x}_i$ equals 1, for all $i$. This corresponds precisely to the "bias assumption" mentioned in § 2. Thus it is now assumed that $X$ can be written as

$$(3.1) \qquad X = \begin{bmatrix} 1 & 1 \cdots 1 \\ \xi_1 & \xi_2 \cdots \xi_m \end{bmatrix} \quad \text{with } \xi_i \in \mathbf{R}^{k-1}.$$

A realization $A = X^TY$ of a given sign matrix will be referred to as an *affine type realization* when $X$ has the form (3.1). Note that the sign matrix $S$ admits an affine type realization of rank $k$ if and only if the extended matrix $[S, \mathbf{s}_0]$ is realizable with the same rank $k$.

Two $k \times m$ matrices are said to be *equivalent* if they allow one to realize the same sets of sign vectors, within signed permutations of the components. Formally, $X'$ is equivalent to $X$ if $\mathscr{S}(X')$ equals $Q\mathscr{S}(X)$ for a certain $m \times m$ matrix $Q$ having a unique element $+$ or $-$ in each row and column, and zero elsewhere. It is clear that any nonzero

$k \times m$ matrix $X$ is equivalent to a matrix of the form (3.1). Note that this concept of equivalence is analogous to "switching equivalence" in graph theory [9].

Our main objective is to characterize the sign vectors $\mathbf{s} = (s_1, s_2, \cdots, s_m)^T$ realizable by means of the matrix (3.1), in terms of the $m$-subset $\mathscr{P}$ of $\mathbf{R}^{k-1}$ defined by

$$\mathscr{P} = \{\xi_1, \xi_2, \cdots, \xi_m\}. \tag{3.2}$$

It will be convenient to specify a given sign vector $\mathbf{s}$ by its two *complementary supports*

$$I^+ = \{i: 1 \leqq i \leqq m, s_i = +\}, \qquad I^- = \{i: 1 \leqq i \leqq m, s_i = -\}. \tag{3.3}$$

This induces a one-to-one correspondence between sign $m$-vectors $\mathbf{s}$ and subsets $I^+$ of $\{1, 2, \cdots, m\}$. From (3.3) let us construct the pair of complementary subsets $\mathscr{P}^+$ and $\mathscr{P}^-$ of $\mathscr{P}$ given by

$$\mathscr{P}^+ = \{\xi_i \in \mathscr{P}: i \in I^+\}, \qquad \mathscr{P}^- = \{\xi_i \in \mathscr{P}: i \in I^-\}. \tag{3.4}$$

Conversely, any subset $\mathscr{P}^+$ of $\mathscr{P}$ yields a sign vector $\mathbf{s} \in \{+, -\}^m$, defined by $s_i = +$ for $\xi_i \in \mathscr{P}^+$ and $s_i = -$ for $\xi_i \in \mathscr{P}^- = \mathscr{P} \setminus \mathscr{P}^+$. In the sequel, $\mathbf{s}$ will be referred to as the *characteristic vector* of $\mathscr{P}^+$ (with respect to a given set $\mathscr{P}$ of cardinality $m$).

To any $m$-subset $\mathscr{P}$ of $\mathbf{R}^{k-1}$ there corresponds a well-defined set of sign $m$-vectors, namely the set $\mathscr{S}(X)$ given by (2.9), via (3.1) and (3.2). We are now in a position to state the following useful version of Theorem 1.

THEOREM 2. *Let $\mathscr{P}^+$ and $\mathscr{P}^-$ be two complementary subsets of a given set $\mathscr{P} = \{\xi_1, \xi_2, \cdots, \xi_m\}$ consisting of $m$ distinct points $\xi_i \in \mathbf{R}^{k-1}$. Then the characteristic vector $\mathbf{s}$ of $\mathscr{P}^+$ with respect to $\mathscr{P}$ belongs to the set $\mathscr{S}(X)$ corresponding to $\mathscr{P}$ if and only if the convex hulls of $\mathscr{P}^+$ and of $\mathscr{P}^-$ are disjoint (in $\mathbf{R}^{k-1}$).*

*Proof*. Assume first that the sign vector $\mathbf{s}$ does not belong to $\mathscr{S}(X)$. By Theorem 1 there exists a linear relation of the form

$$\sum_{i \in I^+} \lambda_i \mathbf{x}_i = \sum_{i \in I^-} \lambda_i \mathbf{x}_i, \tag{3.5}$$

with $\lambda_i \geqq 0$ for all $i \in \{1, 2, \cdots, m\}$, and $\lambda_i > 0$ for at least one index $i$. By equating the first coordinates in both sides of (3.5), and by using (3.1), it is seen that the sums of the $\lambda_i$ over the complementary supports $I^+$ and $I^-$ have the same value. Then the remaining part of (3.5) shows exactly that the convex hull of $\mathscr{P}^+$ and the convex hull of $\mathscr{P}^-$ have a point in common. Conversely, if such is the case, it appears that the convex hull of the $m$ points $s_i \mathbf{x}_i$ of $\mathbf{R}^k$ contains the origin. According to Theorem 1, this means that $\mathbf{s}$ does not belong to the set $\mathscr{S}(X)$, which completes the proof. □

*Remark*. In the nondegenerate case, none of the $k$-subsets of $\mathscr{P}$ is included in an affine hyperplane of $\mathbf{R}^{k-1}$.

The rest of this section contains an application of Theorem 2 for the small values of the dimension $k$ (assumption (3.1) holds throughout). The case $k = 1$ is quite obvious; the only realizable sign vectors are $\mathbf{s} = (+, \cdots, +)^T$ and $\mathbf{s} = (-, \cdots, -)^T$. For $k = 2$, consider an $m$-subset $\mathscr{P}$ of the real line $\mathbf{R}$. The subsets $\mathscr{P}^+$ of $\mathscr{P}$ satisfying the condition of Theorem 2 are those enjoying the property $I^+ < I^-$ or $I^- < I^+$ (elementwise). Hence, $\mathscr{S}(X)$ consists of $2m$ vectors $\mathbf{s}$, given by $\mathbf{s}^T = (\pm)^a(\mp)^b$, with $a + b = m$; the realizable vectors $\mathbf{s}$ are those for which the patterns of plus signs and of minus signs are *linearly contiguous*.

Consider now the case $k = 3$. For a given size $m$, there exist several nonequivalent configurations $\mathscr{P} = \{\xi_1, \xi_2, \cdots, \xi_m\}$ of $m$ distinct points $\xi_i$ in the Euclidean plane $\mathbf{R}^2$. (The relevant concept of equivalence has been defined in the beginning of this section.) The simplest equivalence class contains the *convex configurations* (among others), characterized by the fact that each point $\xi_i$ is outside the convex hull of the remaining $m-1$ points $\xi_j$ of $\mathscr{P}$. An illustration is provided by Fig. 1. The subsets $I^+$ of $I = \{1, 2, \cdots, m\}$ satisfying the condition of Theorem 2 are the $m^2 - m + 2$ "circularly contiguous" sets, of the form $I^+ = \{i_0, i_0+1, \cdots, i_0+t-1\}$, with $0 \leqq t \leqq m$. (An element $i_0 + p$ of $I^+$ is interpreted as the integer $i \in I$ such that $i_0 + p \equiv i$ modulo $m$.) As a result, the realizable vectors $\mathbf{s}$ are those for which the patterns of plus signs and of minus signs are *circularly contiguous*; they are given by $\mathbf{s}^T = (\pm)^a(\mp)^b(\pm)^c$, with $a+b+c=m$. We shall not go into detail about the nonconvex configurations (however, see the final example in this section).

As mentioned in § 2, the $m \times m$ sign matrix $S$ whose entries are

$$s_{i,i} = +, \qquad s_{i,j} = - \quad \text{for } i \neq j \tag{3.6}$$

deserves special attention. It follows from the preceding discussion that $S$ admits an affine type realization matrix $A$ of rank $k=3$. Let us give an explicit solution for this special problem in terms of the vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ that constitute $3 \times m$ matrices $X$ and $Y$ satisfying $\operatorname{sgn}(X^TY) = S$. With a normalization slightly different from that of (3.1), we have the solution

$$\begin{aligned} \mathbf{x}_i &= \left(\cos\frac{\pi}{m}, \cos\frac{2i\pi}{m}, \sin\frac{2i\pi}{m}\right)^T, \\ \mathbf{y}_i &= \left(-\cos\frac{\pi}{m}, \cos\frac{2i\pi}{m}, \sin\frac{2i\pi}{m}\right)^T, \end{aligned} \tag{3.7}$$

for $i = 1, 2, \cdots, m$. Thus the points $\mathbf{x}_i$ and $\mathbf{y}_i$ are regularly distributed on two parallel circles in $\mathbf{R}^3$. Elementary computation shows that the sign condition (2.4) is actually fulfilled. It appears that the rank $k=3$ is the smallest possible in this case (when $m \geqq 4$); this follows from the bound (2.2). Note that the matrix $A = X^TY$ is circulant; a generalization is provided by Theorem 3 in the next section.

Of course, the problem becomes more complicated when the dimension $k$ increases. In the case $k=4$, $m=8$, let us briefly consider the example where $\mathscr{P}$ consists of the

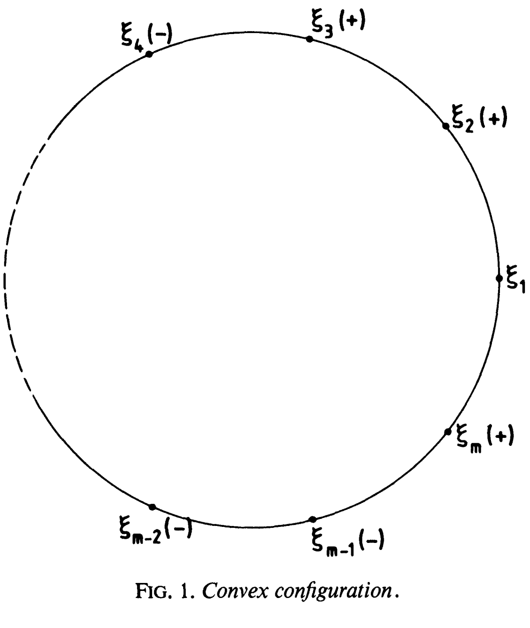


FIG. 1. *Convex configuration.*

eight vertices of a *cube* in the Euclidean space $\mathbf{R}^3$. It can be verified that there are exactly 128 subsets $\mathscr{P}^+$ of $\mathscr{P}$ satisfying the condition of Theorem 2. They consist of the empty set, the eight vertices, the 12 edges, the 24 paths of length two, the three faces containing a given vertex, the four trihedra containing a vertex, the 12 paths of length three starting from a vertex, and the complements of these 64 sets.

As an illustration of the concept of equivalence introduced above, let us finally examine the problem of finding a complete set of nonequivalent $k \times m$ matrices $X$, in the simple case $k = 3$, $m = 6$. It can be seen that there exist four equivalence classes of the nondegenerate type (see the remark after Theorem 2). Besides the class containing the convex configuration (hexagonal, in this example), there are three other classes, which are depicted in Fig. 2. The points $\xi_i$ with $1 \leqq i \leqq 5$ constitute a convex pentagon; the sixth point, $\xi_6$, is denoted by $\alpha$, $\beta$, and $\gamma$ for the three distinct configurations. Let us now indicate half of the subsets $I^+$ of $I = \{1, 2, \cdots, 6\}$ that satisfy the condition of Theorem 2; they are given by

$$\begin{array}{lll} I^+ = \varnothing, & \{1\},\{2\},\{3\},\{4\},\{5\},\{1,2\},\{2,3\}, & \\ & \{3,4\},\{4,5\},\{5,1\},\{1,2,3\},\{1,2,5\},\{1,4,5\}, & \\ \text{and} & \{1,2,6\},\{1,5,6\} & \text{in case } \xi_6 = \alpha, \\ \text{or} & \{3,6\},\{1,2,6\} & \text{in case } \xi_6 = \beta, \\ \text{or} & \{3,6\},\{4,6\} & \text{in case } \xi_6 = \gamma. \end{array} \tag{3.8}$$

The second half consists of the complements of the subsets in (3.8) with respect to the set $I$. Note that the number of realizable sign vectors is 32 in all cases (see Theorem 6 and Corollary 7 in § 5).

**4. Circulants and direct sums.** It seems very difficult to find a general algebraic method to construct a minimum rank realization $A$ of an arbitrary sign matrix $S$. In this section it is explained how some low rank realizations can be explicitly obtained for sign matrices $S$ with interesting special structures.

First assume $S$ is an $n \times n$ *circulant matrix*. (The simplest nontrivial example is provided by (3.6).) It is then natural to examine circulant realization matrices $A$. The

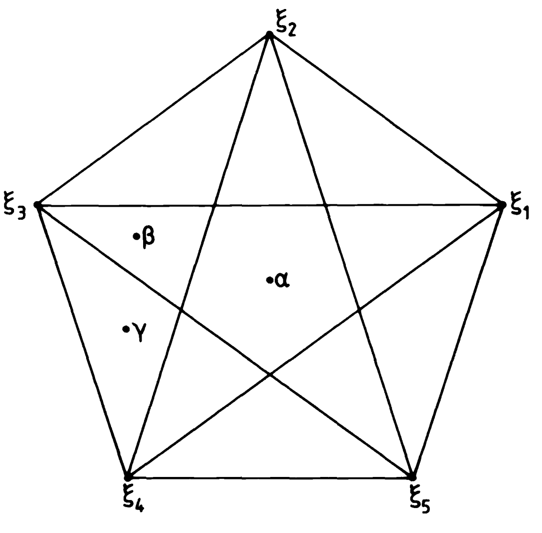


FIG. 2. *Pentagonal configuration.*

problem can be discussed quite easily by means of the cyclic Fourier transform [2]. Define the integer $t$ as the *number of discontinuities* in $S$. This means that there are exactly $t$ minus signs among the products $s_0s_1, \cdots, s_{n-2}s_{n-1}, s_{n-1}s_0$, where $(s_0, s_1, \cdots, s_{n-1})$ is the first row of $S$. (Thus, $s_{i,j} = s_{j-i}$ where $j - i$ is reduced modulo $n$.) Of course, $t$ must be even. The following result is interesting at least when $t/n$ is small.

THEOREM 3. *A circulant sign matrix $S$ with $t$ discontinuities admits an affine type circulant realization matrix $A$ of rank not exceeding $t + 1$.*

*Proof.* Let $(a_0, a_1, \cdots, a_{n-1})$ denote the first row of a real circulant $n \times n$ matrix $A$, and let $(\lambda_0, \lambda_1, \cdots, \lambda_{n-1})$ be its Fourier transform, i.e., the spectrum of $A$. Recall the classical identities

$$\lambda_i = a(\omega^i), \qquad na_j = \lambda(\omega^{-j}), \tag{4.1}$$

for $0 \leqq i, j \leqq n - 1$, where $\omega$ is a fixed primitive $n$th root of unity. Here, $a(z)$ and $\lambda(z)$ denote the polynomials (of formal degree $n - 1$) having $a_i$ and $\lambda_i$ as the coefficients of $z^i$. Note that $\lambda_{n-i}$ equals $\bar{\lambda}_i$, since $A$ is real.

Setting $p = t/2$, consider a real trigonometric polynomial $\lambda(z)$ of the form

$$\lambda(z) = \sum_{i=-p}^{p} \lambda_i z^i, \tag{4.2}$$

with $\lambda_{-i} = \bar{\lambda}_i$. After reduction modulo $z^n - 1$, this can be interpreted, via (4.1), as the Fourier transform of a well-defined real $n$-tuple $(a_0, a_1, \cdots, a_{n-1})$. By construction, the rank of the corresponding circulant matrix $A$ is less than or equal to $2p + 1$.

Let us now examine the sign condition (2.1), expressed by $\operatorname{sgn}(\lambda(\omega^{-j})) = s_j$ for $j = 0, 1, \cdots, n - 1$ in view of (4.1). This requirement will be fulfilled by any real trigonometric polynomial (4.2) having one zero on each of the $t$ arcs of the unit circle where a sign discontinuity occurs, and satisfying $\operatorname{sgn}(\lambda(1)) = s_0$. Thus, $\lambda(z)$ can be written as

$$\lambda(z) = cz^{-p} \prod_{i=1}^{t} \frac{z - \zeta_i}{1 - \zeta_i}, \tag{4.3}$$

with $\operatorname{sgn}(c) = s_0$, for an appropriate choice of $t$ complex numbers $\zeta_i$ of unit modulus. Further details concerning the sign property (2.1) are omitted.

Finally, it can easily be verified that $A$ is an affine type realization (provided the zeros $\zeta_i$ are chosen properly). To see this, consider the spectral decomposition

$$A = U\Lambda U^T, \tag{4.4}$$

where $U$ is a well-defined $n \times (1 + 2p)$ submatrix of the real version of the $n \times n$ Fourier transform matrix. (It satisfies $U^TU = I$.) By construction, the matrix $\Lambda$ in (4.4) is block-diagonal; it has one diagonal element equal to the real eigenvalue $\lambda_0$, and $p$ real diagonal blocks of order 2 admitting the pairs of complex conjugate eigenvalues $(\lambda_i, \lambda_{-i})$ for $i = 1, 2, \cdots, p$. In view of (2.3), the result (4.4) provides the solution $X = U^T$, $Y = \Lambda U^T$. The eigenvector of $A$ corresponding to the "principal eigenvalue" $\lambda_0 = a(1)$ is the all-one vector. Therefore, the first row of $X$ is a multiple of $(1, 1, \cdots, 1)$ provided the coefficient $\lambda_0$ of $\lambda(z)$ does not vanish. This requirement can be fulfilled by a suitable choice of the zeros $\zeta_i$. $\square$

As a second theme, we now consider a kind of "direct sum construction." For an integer $q \geqq 2$, let there be given any $q$-tuple of sign matrices $S_1, S_2, \cdots, S_q$, of respective sizes $m_1 \times n_1, m_2 \times n_2, \cdots, m_q \times n_q$. Construct the $m \times n$ sign matrix

$$S = \begin{bmatrix} S_1 & - & \cdots & - \\ - & S_2 & \cdots & - \\ \vdots & \vdots & & \vdots \\ - & - & \cdots & S_q \end{bmatrix}, \tag{4.5}$$

with $m = m_1 + \cdots + m_q$ and $n = n_1 + \cdots + n_q$. This simply means that each entry of $S$ outside the diagonal blocks $S_i$ is a minus sign. By abuse of terminology, $S$ will be referred to as the *direct sum* of $S_1, S_2, \cdots, S_q$. With the help of Theorem 2 we can establish the following remarkable result.

THEOREM 4. *If all sign matrices $S_1, S_2, \cdots, S_q$ admit affine type realizations of rank not exceeding $k$, for a given integer $k \geqq 2$, then their direct sum $S$ admits an affine type realization of rank not exceeding $k + 1$.*

*Proof.* Let $\Omega = \{\mathbf{y} \in \mathbf{R}^k: \mathbf{y}^T\mathbf{y} = 1\}$ denote the $(k-1)$-dimensional unit sphere in the Euclidean space $\mathbf{R}^k$. Choose $q$ distinct points $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_q$ on $\Omega$. For $i = 1, 2, \cdots, q$, define $H_i$ to be the *tangent hyperplane* to the sphere $\Omega$ at the point $\mathbf{w}_i$. In the $(k-1)$-dimensional affine space $H_i$, construct a configuration $\mathcal{P}_i$ of $m_i$ points,

$$\mathcal{P}_i = \{\xi_{i,1}, \xi_{i,2}, \cdots, \xi_{i,m_i}\} \subset H_i, \tag{4.6}$$

that allows us to realize each of the $n_i$ columns of the sign matrix $S_i$. According to Theorem 2, this means that the convex hulls of two complementary subsets $\mathcal{P}_i^+$ and $\mathcal{P}_i^-$ of $\mathcal{P}_i$ are disjoint (in $H_i$), for each $\mathcal{P}_i^+$ whose characteristic vector is a column of $S_i$.

Next, let $C_i$ denote the convex hull of the union of the $q-1$ sets $\mathcal{P}_j$ with $j \neq i$, and define $\hat{C}_i$ to be the *pointed convex cone* of vertex $\mathbf{w}_i$ based on $C_i$. More precisely, $\hat{C}_i$ consists of the points $\mathbf{w}_i + \rho(\mathbf{y} - \mathbf{w}_i)$, with $\mathbf{y} \in C_i$ and $\rho \in \mathbf{R}^+$. An illustration is provided by Fig. 3. In the sequel it is assumed that all points $\xi_{i,t}$ of $\mathcal{P}_i$ are sufficiently close to the

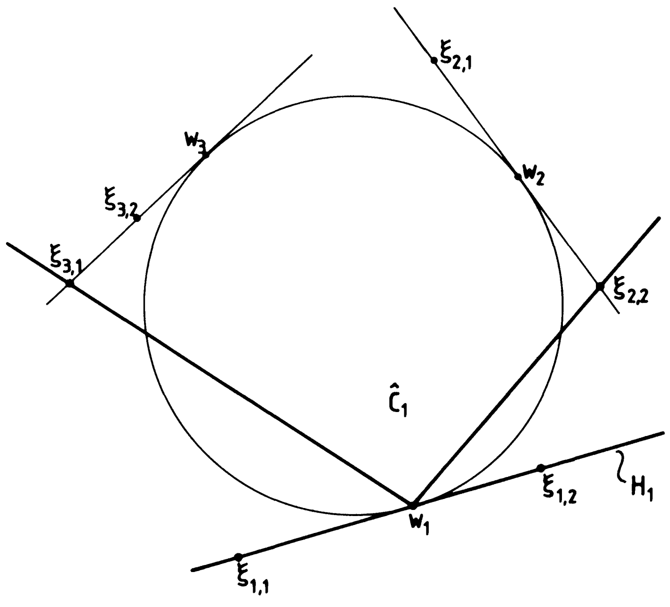


FIG. 3. *Direct sum construction.*

contact point $\mathbf{w}_i$, so as to ensure the property

$$\hat{C}_i \cap H_i = \{\mathbf{w}_i\}. \tag{4.7}$$

To prove the theorem it suffices to show that the direct sum matrix $S$ is realizable by means of the $m$-subset $\mathcal{P}$ of $\mathbf{R}^k$ defined as the union configuration

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \cup \mathcal{P}_q. \tag{4.8}$$

For given integers $i$ and $l$, with $1 \leqq i \leqq q$ and $1 \leqq l \leqq n_i$, let $I^+$ and $I^-$ denote the complementary supports of the $l$th column of $S_i$. According to Theorem 2, we have only to verify that the convex hull of the subset $\mathcal{P}^+ = \mathcal{P}_i^+ = \{\xi_{i,t}: t \in I^+\}$ of $\mathcal{P}$ is disjoint from the convex hull of the complementary subset $\mathcal{P}^- = \mathcal{P} \backslash \mathcal{P}^+$ (for all choices of $i$ and $l$).

Suppose this is not true. Then there exists a linear relation

$$\sum_{t \in I^+} \lambda_{i,t}\xi_{i,t} = \sum_{t \in I^-} \lambda_{i,t}\xi_{i,t} + \sum_{j \neq i} \sum_{t=1}^{m_j} \lambda_{j,t}\xi_{j,t}, \tag{4.9}$$

where the coefficients $\lambda$ are nonnegative and add up to unity on both sides. By subtracting $\mathbf{w}_i$ we obtain

$$\sum_{t \in I^+} \lambda_{i,t}(\xi_{i,t} - \mathbf{w}_i) - \sum_{t \in I^-} \lambda_{i,t}(\xi_{i,t} - \mathbf{w}_i) = \sum_{j \neq i} \sum_{t=1}^{m_j} \lambda_{j,t}(\xi_{j,t} - \mathbf{w}_i). \tag{4.10}$$

The left-hand side belongs to the hyperplane $H_i - \mathbf{w}_i$ while the right-hand side belongs to the convex cone $\hat{C}_i - \mathbf{w}_i$ (with vertex $\mathbf{0}$). In view of (4.7), the only possibility for the common point in (4.10) is the origin. This forces $\lambda_{j,t} = 0$ for all $j \neq i$ and $1 \leqq t \leqq m_j$. As a consequence, we deduce the relation

$$\sum_{t \in I^+} \lambda_{i,t}\xi_{i,t} = \sum_{t \in I^-} \lambda_{i,t}\xi_{i,t}, \tag{4.11}$$

where the coefficients add up to unity on both sides. This contradicts the fact that the convex hulls of $\mathcal{P}_i^+$ and of $\mathcal{P}_i^-$ are disjoint. Hence the theorem is proved. □

*Remark.* In fact, the method described in the proof allows us to construct an $m \times n$ realization matrix $A$ of rank $\leqq k + 1$ whose diagonal blocks coincide with any *given* realization matrices $A_1, A_2, \cdots, A_q$, of rank $\leqq k$, of the sign blocks $S_1, S_2, \cdots, S_q$.

Applying Theorem 4 to the elementary case $m_i = n_i = 1$ for all $i$, with $k = 2$, shows (once again) that the sign matrix $S$ in (3.6) admits an affine type realization of rank 3. Let us now mention an interesting more general application of Theorem 4. Consider a sign matrix $S$ having at most two plus signs in each row and each column. Such a matrix will be said to have *degree* 2. Within permutations of rows and columns, $S$ can be represented as a direct sum of matrices $S_1, S_2, \cdots, S_q$, where each $S_i$ is a contiguous submatrix of a circulant having $(+, +, -, \cdots, -)$ as its first row. (This representation is obtained by constructing the set of row-column paths of plus signs in $S$.) As explained in § 3 (about "convex configurations"), and as shown by Theorem 3, the matrices $S_i$ of that type admit affine type realizations of rank $\leqq 3$. Hence Theorem 4 contains the following result as a special case.

COROLLARY 5. *Any sign matrix of degree* 2 *admits an affine type realization of rank not exceeding* 4.

**5. Counting realizable sign vectors.** This section is devoted to an analysis of the set $\mathcal{S}(X)$ of the sign vectors $\mathbf{s} \in \{+, -\}^m$ realizable by means of a given $k \times m$ matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]$, with $\mathbf{x}_i \in \mathbf{R}^k$ (see (2.9)). Two questions arise naturally in that context. First, *what is the cardinality of* $\mathscr{S}(X)$? The answer will give the maximum number of distinct sign patterns among the columns of an $m \times n$ matrix of rank $k$. Second, given positive integers $k$ and $m$, *in how many equivalence classes does the whole set of* $k \times m$ *matrices* $X$ *partition*? The relevant concept of equivalence has been introduced and illustrated in § 3.

We shall only examine the former problem, also treated implicitly in [1]; of course the latter problem is considerably more difficult. In the sequel we assume that the matrix $X$ is *nondegenerate*, in the sense that any $p$-tuple of columns of $X$ is a basis of the space $\mathbf{R}^p$ when $p \leqq k$. (For a technical reason it is sometimes useful to consider the "defective case" where $m$ is less than $k$.) As shown below, under this assumption the cardinality of $\mathscr{S}(X)$ depends only on the parameters $k$ and $m$, not on the entries of the matrix $X$. (Degenerate matrices $X$ yield smaller sets $\mathscr{S}(X)$.)

From a given nondegenerate $k \times m$ matrix $X = [\mathbf{x}_1, \cdots, \mathbf{x}_m]$ we construct a *spherical polytope P*, on the $(k-1)$-dimensional unit sphere $\Omega = \{\mathbf{y} \in \mathbf{R}^k \colon \mathbf{y}^T\mathbf{y} = 1\}$, by taking the $m$ hyperplanes $\mathbf{x}_i^T\mathbf{y} = 0$, for $i = 1, 2, \cdots, m$, as boundaries. It possesses spherical *faces* of dimension $0, 1, \cdots, k-1$, defined in essentially the same way as in the classical theory of convex polytopes [10]. A $(k-t)$-dimensional face of $P$ is a subpolytope on a sphere defined by the $t$ equations

$$\mathbf{y}^T\mathbf{y} = 1, \qquad \mathbf{x}_{i_1}^T\mathbf{y} = \mathbf{x}_{i_2}^T\mathbf{y} = \cdots = \mathbf{x}_{i_{t-1}}^T\mathbf{y} = 0, \tag{5.1}$$

for a given choice of $t-1$ distinct integers $i_1, i_2, \cdots, i_{t-1}$ among $1, 2, \cdots, m$.

The zero-dimensional faces are the *vertices* of $P$. Each vertex is an isolated point $\mathbf{y}$ obtained as the solution of a system of equations (5.1), with $t = k$. Note that $P$ is antipodal, in the sense that the vertices occur in opposite pairs $(\mathbf{y}, -\mathbf{y})$. (Of course, there exist vertices only in the case $m \geqq k-1$.) The $(k-1)$-dimensional faces are referred to as the *facets* of $P$; they are the connected regions of $\Omega$ separated by the boundary hyperplanes $\mathbf{x}_i^T\mathbf{y} = 0$. It is easily seen that the facets are in one-to-one correspondence with the elements $\mathbf{s}$ of the set $\mathscr{S}(X)$. More precisely, the interior of a given facet is nothing but a region of $\Omega$ yielding a fixed sign vector $\mathbf{s} = \operatorname{sgn}(X^T\mathbf{y})$. An illustration is provided by Fig. 4 in the very simple case $k = 2$, $m = 4$. The plus and minus signs indicate the regions of the plane where the inner products $\mathbf{x}_i^T\mathbf{y}$ are positive and negative (for $i = 1, 2, 3, 4$). The correspondence between the facets, which are circular arcs, and the sign vectors is obvious.

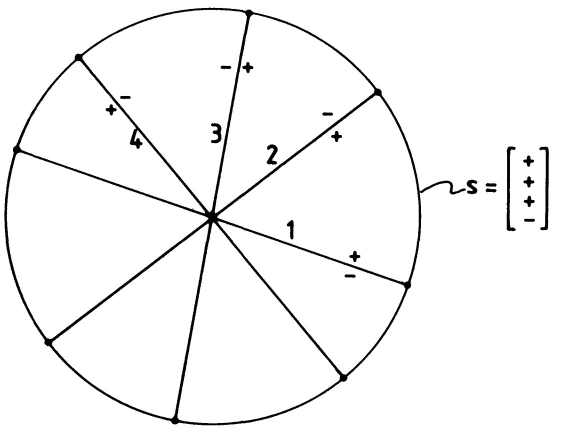


FIG. 4. *Example with* $k = 2$, $m = 4$.

Our problem is now to count the facets of $P$. It proves useful to extend this to the question of determining the number of faces of each possible dimension (between $\max(0, k-m-1)$ and $k-1$). Indeed, as shown below, the extended problem can be solved explicitly with the help of a simple recurrence relation. Let us denote by $f_j(k, m)$ the number of $j$-dimensional faces of the spherical polytope $P$, for a given $k \times m$ nondegenerate matrix $X$. We shall see that this number depends only on $j$, $k$ and $m$; we shall obtain the explicit formula

$$f_{k-t}(k,m) = 2\binom{m}{t-1}\sum_{i=0}^{k-t}\binom{m-t}{i}, \tag{5.2}$$

for $t = 1, 2, \cdots, k$, in terms of binomial coefficients.

Note that (5.2) is obvious in the case $t = k$. Indeed, the number of vertices of $P$ is given by

$$f_0(k,m) = 2\binom{m}{k-1}, \tag{5.3}$$

as it follows from definition (5.1) with $t = k$. The other extreme case, $t = 1$, yields the solution to our main problem; the number of facets of $P$ is given by

$$f_{k-1}(k,m) = 2\sum_{i=0}^{k-1}\binom{m-1}{i}. \tag{5.4}$$

Before proving the general result (5.2) let us explain why formula (5.4) seems rather plausible a priori. A simple argument yields the identity

$$f_{k-1}(k,m) = 2^m \quad \text{when } m = 1, 2, \cdots, k. \tag{5.5}$$

(This is the "defective case," for which the problem is elementary.) Furthermore, it is a plausible conjecture that $f_j(k, m)$ can be expressed as a polynomial of degree $k-1$ in the variable $m$ (as suggested by (5.3), in particular). Formula (5.4) follows immediately from this conjecture, since its right-hand side is the unique polynomial of degree $k-1$ satisfying the interpolation constraint (5.5).

To establish (5.2) we shall make use of two simple identities. The first one is the *Euler–Poincaré formula*

$$\sum_{t=1}^{k}(-1)^t f_{k-t}(k,m) = (-1)^k - 1. \tag{5.6}$$

This is a well-known result in the theory of convex polytopes (see [10, p. 78]). It expresses the fact that the Möbius function of the face lattice assumes alternatively the values 1 and $-1$ (see [5]), which is true for spherical polytopes as well as for classical convex polytopes. The second key identity is the *recurrence relation*

$$(t-1)f_{k-t}(k,m) = m f_{k-t}(k-1, m-1), \tag{5.7}$$

for $t = 2, 3, \cdots, k$. A proof will be given below.

We can deduce (5.2) from (5.6) and (5.7) in a very economical manner as follows. All numbers $f_j(k, m)$ are computable by use of (5.6) and (5.7) from the initial condition $f_0(1, m) = 2$. This method produces $f_j(k, m)$ as a polynomial of degree $k-1$ in $m$. As explained above, the defective case identity (5.5) forces the result (5.4), whence the general formula (5.2) by iterative application of (5.7). Note that, in this argument, the identity (5.6) is used only to ensure the desired degree property of $f_{k-1}(k, m)$; substituting the values (5.2) thus obtained, we can check that the Euler–Poincaré formula (5.6) is actually satisfied.

We still must prove the recurrence relation (5.7). The argument uses an induction over the dimension $k$; it is assumed that the number $f_j(k-1, m-1)$ is independent of the underlying $(k-1) \times (m-1)$ matrix $X'$, for $j = 0, 1, \cdots, k-2$. Let us compute, in two different manners, the number $N$ counting the pairs $(F, H)$ where $F$ is a $(k-t)$-dimensional face of $P$, for a fixed $t$ with $2 \leqq t \leqq k$, and $H$ is a boundary hyperplane of $P$ that contains $F$. A given face $F$ spans a $(k+1-t)$-dimensional subspace of $\mathbf{R}^k$, defined as the intersection of $t-1$ hyperplanes $\mathbf{x}_i^T\mathbf{y} = 0$; this yields the expression $N = (t-1)f_{k-t}(k, m)$. Next, for a given hyperplane $H$ (defined by $\mathbf{x}_i^T\mathbf{y} = 0$ for a certain $i$), we can interpret the intersection $P' = P \cap H$ as a spherical polytope, on the $(k-2)$-dimensional unit sphere $\Omega' = \Omega \cap H$, corresponding to a well-defined nondegenerate $(k-1) \times (m-1)$ matrix $X'$. As a consequence, by use of the induction assumption, we obtain the formula $N = mf_{k-t}(k-1, m-1)$. By equating both expressions of $N$ we get the desired result (5.7). When combined with (5.6), this shows that $f_j(k, m)$ is actually independent of the matrix $X$, for $j = 0, 1, \cdots, k-1$.

Let us now state the main conclusion of our analysis; an equivalent result, with a different proof, can be found in a recent paper by Abu-Mostafa and St. Jacques [1].

THEOREM 6. *For a nondegenerate $k \times m$ matrix $X$, the cardinality of the set $\mathcal{S}(X)$ of sign vectors realizable by means of $X$ is given by*

$$|\mathcal{S}(X)| = f_{k-1}(k,m) = 2\sum_{i=0}^{k-1}\binom{m-1}{i}. \tag{5.8}$$

The examples given in § 3 agree with this result. Note the consistency of (5.8) with the lower bound (2.2) on the realizable rank. Indeed, if $\mathcal{S}(X)$ is the whole set $\{+, -\}^m$, i.e., if $|\mathcal{S}(X)|$ equals $2^m$, then (5.8) implies $k \geqq m$. Let us finally emphasize the interpretation of Theorem 6 in the context of § 3 (see Theorem 2).

COROLLARY 7. *Let $\mathcal{P}$ by any m-subset of the Euclidean space $\mathbf{R}^{k-1}$ having the property that no k-subset of $\mathcal{P}$ is included in an affine hyperplane. Define g to be the number of subsets $\mathcal{P}^+$ of $\mathcal{P}$ for which the convex hull of $\mathcal{P}^+$ is disjoint from the convex hull of the complementary subset $\mathcal{P}^- = \mathcal{P} \setminus \mathcal{P}^+$. This number g depends only on k and m (not on $\mathcal{P}$ itself); it is given by $g = f_{k-1}(k, m)$.*

## REFERENCES

[1] Y. S. ABU-MOSTAFA AND J.-M. ST. JACQUES, *Information capacity of the Hopfield model*, IEEE Trans. Inform. Theory, 31 (1985), pp. 461–464.

[2] P. J. DAVIS, *Circulant Matrices*, John Wiley, New York, 1979.

[3] J. J. HOPFIELD, *Neural networks and physical systems with emergent collective computational abilities*, Proc. Nat. Acad. Sci. U.S.A., 79 (1982), pp. 2554–2558.

[4] R. P. LIPPMANN, *An introduction to computing with neural nets*, IEEE ASSP Magazine, April 1987, pp. 4–22.

[5] G.-C. ROTA, *On the combinatorics of the Euler characteristic*, in Studies in Pure Mathematics, The R. Rado Festschrift, L. Mirsky, ed., Academic Press, New York, 1971, pp. 221–234.

[6] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning internal representations by error propagation*, in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, MA, 1986, pp. 318–362.

[7] D. E. RUMELHART AND J. L. MCCLELLAND, EDS., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vols. 1 and 2, MIT Press, Cambridge, MA, 1986.

[8] A. SCHRIJVER, *Theory of Linear and Integer Programming*, John Wiley, New York, 1986.

[9] J. J. SEIDEL, *A survey of two-graphs*, in Proc. Int. Coll. Teorie Combinatorie, Acc. Naz. Lincei, Roma, 1976, pp. 481–511.

[10] J. STOER AND C. WITZGALL, *Convexity and Optimization in Finite Dimensions* I, Springer-Verlag, Berlin, 1970.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 64-67, February 1989

# A NOTE ON THE VERTEX ARBORICITY OF A GRAPH*

S. L. HAKIMI† AND E. F. SCHMEICHEL‡

**Abstract.** The *vertex arboricity* $a(G)$ of a graph $G$ is the minimum number of subsets into which the vertices of $G$ can be partitioned so that each subset induces an acyclic graph. A characterization of planar graphs $G$ is given for which $a(G) = 2$, thereby answering a question of Grünbaum [*Israel J. Math.*, 14 (1973), pp. 390–408]. The characterization is in terms of the dual graph $G^*$. As a corollary, a theorem of Stein that characterizes maximal planar graphs $G$ with $a(G) = 2$ is obtained. This latter result implies that determining whether $a(G) \leqq 2$ is NP-complete for maximal planar graphs $G$.

**Key words.** vertex arboricity, NP-completeness, Hamiltonian graph, dual graph, edge arboricity

**AMS(MOS) subject classifications.** 05C45, 05C99, 68C25

**1. Introduction.** Our terminology and notation will be standard except as indicated. Good references for any undefined terms are [1] and [2].

The *vertex arboricity* of an undirected graph $G$, denoted $a(G)$, is the minimum number of subsets into which $V(G)$ can be partitioned so that each subset induces an acyclic graph. Our purpose here is to give a characterization of planar graphs $G$ for which $a(G) = 2$, thereby answering a question of Grünbaum [6, p. 404]. The characterization is in terms of the dual graph $G^*$. Stein [11] previously characterized maximal planar graphs $G$ satisfying $a(G) = 2$. Stein's characterization follows easily from ours, and our approach, which is completely different from Stein's, appears to be more elementary.

These characterizations have a number of interesting applications. It was known previously [4, p. 193] that determining the vertex arboricity of a graph is NP-hard. We will show that determining whether $a(G) \leqq 2$ is NP-complete for maximal planar graphs $G$. These characterizations also imply a classic theorem of Tait [12], and provide an interesting reformulation of a well-known conjecture of Barnette [2, p. 248]. In addition, we give an upper bound for $a(G)$ analogous to the upper bound of Welsh and Powell [14] for the chromatic number of $G$. We conclude by briefly discussing the related parameter of edge arboricity.

**2. Main results.** It is easy to show by induction that $a(G) \leqq 3$ for any planar graph $G$. We begin by characterizing planar graphs $G$ with $a(G) = 2$ in terms of their dual graph $G^*$.

THEOREM 1. *Let $G$ be a planar graph. Then $a(G) = 2$ if and only if $G^*$ contains a connected Eulerian spanning subgraph.*

*Proof.* Suppose that $a(G) = 2$. Let $\{V_1, V_2\}$ be an acyclic partition of $V(G)$ (i.e., the graph induced by $V_i$ is acyclic, for $i = 1, 2$). Let $E(V_1, V_2)$ denote the edges in $G$ joining a vertex in $V_1$ to one in $V_2$, and consider the corresponding set of edges $E'$ in $G^*$. Let $H$ denote the subgraph of $G^*$ induced by $E'$; we will show that $H$ is a connected Eulerian (i.e., every vertex has even degree) spanning subgraph of $G^*$.

Since $E(V_1, V_2)$ is an edge cut in $G$, the graph $H$ is Eulerian. Since every cycle of $G$ contains an edge of $E(V_1, V_2)$, every facial cycle of $G$ contains one or more edges of

* Received by the editors January 18, 1988; accepted for publication June 13, 1988. This work was supported by the National Science Foundation under grants ECS-8700656 and ASC-8740647.

† Department of Electrical Engineering and Computer Science, University of California, Davis, California 95616.

‡ Department of Mathematics and Computer Science, San Jose State University, San Jose, California 95192.

$E(V_1, V_2)$, and hence $H$ is spanning in $G^*$. If $H$ were disconnected, then $G^*$ contains an edge cut $E'_1$ containing none of the edges of $E'$. But then the corresponding set of edges $E_1$ in $G$ would induce a Eulerian subgraph $G_1$ in $G$ containing none of the edges in $E(V_1, V_2)$. Thus $G$ contains a cycle including none of the edges in $E(V_1, V_2)$, contradicting the assumption that $\{V_1, V_2\}$ is an acyclic partition in $G$.

Conversely, suppose $G^*$ contains a connected Eulerian spanning subgraph $H'$. Let $H$ denote the subgraph induced by the corresponding set of edges in $G$. Since $H'$ is Eulerian, the edges of $H$ form an edge cut $E(V_1, V_2)$ in $G$. Since every edge cut in $G^*$ contains at least one edge of $H'$, every cycle in $G$ contains one or more edges of $E(V_1, V_2)$. Thus the graph induced by $V_i$ is acyclic for $i = 1, 2$, and so $a(G) = 2$. □

We now easily obtain the following result of Stein [11, Thm. 2.2].

THEOREM 2. *Let $G$ be a maximal planar graph on four or more vertices. Then $a(G) = 2$ if and only if $G^*$ is Hamiltonian.*

*Proof.* Since $G$ is maximal planar, the dual graph $G^*$ is cubic. Simply note that a connected Eulerian spanning subgraph in a cubic graph is a Hamiltonian cycle. □

Now consider the following problem.

VERTEX PARTITION INTO TWO FORESTS

Input: An undirected graph $G = (V, E)$

Question: Does $V$ have a partition $\{V_1, V_2\}$ such that the graph induced by $V_i$ is acyclic for $i = 1, 2$?

The authors showed previously [7] that VERTEX PARTITION INTO TWO FORESTS is NP-complete for general graphs $G$ (reduction from NOT-ALL-EQUAL 3-SAT [4, p. 259]). Using Theorem 2, we obtain the following stronger result.

THEOREM 3. *VERTEX PARTITION INTO TWO FORESTS is NP-complete for maximal planar graphs.*

*Proof.* Reduction from HAMILTONIAN CIRCUIT for planar cubic 3-connected graphs [5]. Simply note that the dual of any such graph is a 3-connected maximal planar graph, and use Theorem 2. □

By standard arguments [4, pp. 141–142], Theorem 3 implies that there does not exist a polynomial time approximation algorithm for vertex arboricity that always partitions $V$ into fewer than $\frac{3}{2}a(G)$ subsets, each inducing an acyclic graph, unless $P = NP$.

Theorem 2 also yields the following classic result [12].

TAIT'S THEOREM. *Let $G$ be a maximal planar graph. If $G^*$ is Hamiltonian, then $G$ is 4-colorable.*

*Proof.* If $a(G) = 2$, then $G$ is 4-colorable. □

The dual of a maximal planar graph on four or more vertices is a planar cubic 3-connected graph. Tait conjectured that every planar cubic 3-connected graph is Hamiltonian. If Tait's conjecture were true, the above theorem would have settled the famous Four Color Conjecture. But Tutte [13] showed that not every planar cubic 3-connected graph is Hamiltonian. However, the following weaker conjecture of Barnette [2, p. 248] remains open.

BARNETTE'S CONJECTURE. *Every bipartite planar cubic 3-connected graph is Hamiltonian.*

Using Theorem 2, we can reformulate Barnette's conjecture in dual form as follows.

BARNETTE'S CONJECTURE (Dual Form). *If $G$ is an Eulerian maximal planar graph (equivalently [2, p. 159], if $G$ is a 3-colorable, maximal planar graph), then $a(G) = 2$.*

The authors are not aware of any 3-colorable planar graph $G$ with $a(G) > 2$. The nonexistence of such a graph would, of course, be a generalization of Barnette's Conjecture.

We next establish an upper bound on $a(G)$ in terms of the vertex degrees of $G$. The result is analogous to an upper bound for the chromatic number of $G$ due to Welsh and Powell [14].

THEOREM 4. *Let $G = (V, E)$ be a graph with vertices $\{v_1, v_2, \cdots, v_n\}$. Let $d_i$ denote the degree of $v_i$ in $G$, and assume that $d_1 \geqq d_2 \geqq \cdots \geqq d_n$. Then*

$$a(G) \leqq \max_{1 \leqq i \leqq n} \min \left\{ \left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{d_i+1}{2} \right\rceil \right\}.$$

*Proof.* The result is trivial for small $n$, and we proceed by induction. Let $V_1$ be any maximal set of vertices containing $v_1$ and $v_2$ and such that the graph induced by $V_1$ is acyclic. The maximality of $V_1$ implies that any vertex $u \in V - V_1$ must be adjacent to at least two vertices in $V_1$.

Let $G' = G - V_1$ contain the vertices $\{w_1, w_2, \cdots, w_{n'}\}$, where $n' \leqq n - 2$. Let $D_i$ denote the degree of $w_i$ in $G'$, and suppose that $D_1 \geqq D_2 \geqq \cdots \geqq D_{n'}$. By the induction hypothesis,

$$(*) \qquad a(G') = \max_{1 \leqq i \leqq n'} \min \left\{ \left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{D_i+1}{2} \right\rceil \right\}.$$

Suppose a maximum for the right side of $(*)$ occurs at $i = i_0$. Then $d_{i_0+2} \geqq D_{i_0} + 2$, and so we have

$$a(G) \leqq a(G') + 1 \leqq \min \left\{ \left\lceil \frac{i_0}{2} \right\rceil, \left\lceil \frac{D_{i_0}+1}{2} \right\rceil \right\} + 1 = \min \left\{ \left\lceil \frac{i_0+2}{2} \right\rceil, \left\lceil \frac{(D_{i_0}+2)+1}{2} \right\rceil \right\}$$

$$\leqq \min \left\{ \left\lceil \frac{i_0+2}{2} \right\rceil, \left\lceil \frac{d_{i_0+2}+1}{2} \right\rceil \right\} \leqq \max_{1 \leqq i \leqq n} \min \left\{ \left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{d_i+1}{2} \right\rceil \right\}$$

as desired. □

In conclusion, let us briefly mention the *edge arboricity* of a graph $G$, denoted $a_e(G)$, which is the minimum number of subsets into which the edges of $G$ can be partitioned so that each subset is an acyclic graph. We mention first that there is an elegant formula for $a_e(G)$ due to Nash-Williams [10]; no analogous formula for $a(G)$ is known.

THEOREM. *For any graph $G$, $a_e(G) = \max \lfloor |E(H)|/(|V(H)| - 1) \rfloor$, where the maximum is taken over all nontrivial induced subgraphs $H$ of $G$.*

We also mention that $a_e(G)$ (in contrast to $a(G)$) can be determined in polynomial time using the matroid partitioning algorithm of Edmonds [3]. (See also [8] and [9, p. 320].)

## REFERENCES

[1] M. BEZHAD, G. CHARTRAND, AND L. LESNIAK-FOSTER, *Graphs and Digraphs*, Wadsworth, Belmont, 1979.
[2] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, North-Holland, New York, 1976.
[3] J. EDMONDS, *Minimum partition of a matroid into independent subsets*, J. Res. Nat. Bur. Standards, 69 B (1965), pp. 147–153.
[4] M. GAREY AND D. JOHNSON, *Computers and Intractability*, W. H. Freeman, San Francisco, 1979.
[5] M. GAREY, D. JOHNSON, AND R. E. TARJAN, *The planar Hamiltonian circuit problem is* NP-*complete*, SIAM J. Comput., 5 (1976), pp. 704–714.
[6] B. GRÜNBAUM, *Acyclic colorings of planar graphs*, Israel J. Math., 14 (1973), pp. 390–408.
[7] S. L. HAKIMI AND E. F. SCHMEICHEL, unpublished manuscript.

[8] T. KAMEDA, *On maximally distant spanning trees of a graph*, Computing, 17 (1976), pp. 115–119.
[9] E. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.
[10] C. ST. J. A. NASH-WILLIAMS, *Decomposition of a finite graph into forests*, J. London Math. Soc., 39 (1964), p. 12.
[11] S. K. STEIN, *B-sets and planar maps*, Pacific J. Math., 37 (1971), pp. 217–224.
[12] P. G. TAIT, *Remarks on colouring of maps*, Proc. Roy. Soc. Edinburgh, 10 (1880), p. 729.
[13] W. T. TUTTE, *On Hamilton circuits*, J. London Math. Soc., 21 (1946), pp. 98–101.
[14] D. WELSH AND M. POWELL, *An upper bound for the chromatic number of a graph and its application to timetabling problems*, Comput. J., 10 (1967), pp. 85–87.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 68-72, February 1989

# ON THE SIZE OF SYSTEMS OF SETS EVERY $t$ OF WHICH HAVE AN SDR, WITH AN APPLICATION TO THE WORST-CASE RATIO OF HEURISTICS FOR PACKING PROBLEMS*

C. A. J. HURKENS† AND A. SCHRIJVER‡

**Abstract.** Let $E_1, \cdots, E_m$ be subsets of a set $V$ of size $n$, such that each element of $V$ is in at most $k$ of the $E_i$ and such that each collection of $t$ sets from $E_1, \cdots, E_m$ has a system of distinct representatives (SDR). It is shown that $m/n \leqq (k(k-1)^r - k)/(2(k-1)^r - k)$ if $t = 2r - 1$, and $m/n \leqq (k(k-1)^r - 2)/(2(k-1)^r - 2)$ if $t = 2r$. Moreover it is shown that these upper bounds are the best possible. From these results the "worst-case ratio" of certain heuristics for the problem of finding a maximum collection of pairwise disjoint sets among a given collection of sets of size $k$ is derived.

**Key words.** packing, system of distinct representatives, worst-case ratio, heuristics

**AMS(MOS) subject classifications.** 05C65, 05A05, 90C27

## 1. Introduction.

We prove the following theorem, where $m$, $n$, $k$, and $t$ are positive integers, with $k \geqq 3$.

THEOREM 1. *Let $E_1, \cdots, E_m$ be subsets of the set $V$ of size $n$, such that we have the following*:

(1) (i) *Each element of $V$ is contained in at most $k$ of the sets $E_1, \cdots, E_m$*;
(ii) *Any collection of at most $t$ sets among $E_1, \cdots, E_m$ has a system of distinct representatives.*

*Then, we have the following*:

$$(2)\quad \text{(i)}\ \frac{m}{n} \leqq \frac{k(k-1)^r - k}{2(k-1)^r - k} \quad \text{if } t = 2r - 1;$$

$$\text{(ii)}\ \frac{m}{n} \leqq \frac{k(k-1)^r - 2}{2(k-1)^r - 2} \quad \text{if } t = 2r.$$

Note that by the König-Hall Theorem, condition (1)(ii) can be replaced by the following:

(3) For any $s \leqq t$, any $s$ of the sets among $E_1, \cdots, E_m$ cover at least $s$ elements of $V$.

We give a proof of Theorem 1 in § 2. We also show that the bounds given in (2) are best possible in the following sense.

THEOREM 2. *For any fixed $k$, $t$ (with $k \geqq 3$), there exist $m$, $n$ and $E_1, \cdots, E_m \subseteq V$ (with $|V| = n$) satisfying* (1) *and having equality in the appropriate line of* (2).

The proof of Theorem 2 is based on a construction using regular graphs of large girth (see § 3).

Finally, in § 4 we apply these results to derive the worst-case ratio of certain heuristic algorithms for the problem of finding a largest family of pairwise disjoint sets among a given family of sets of size $k$ (this problem is NP-complete for any $k \geqq 3$).

* Received by the editors April 1, 1988; accepted for publication July 13, 1988. Part of this research was performed while the authors were visiting the Rutgers Center for Operations Research, Rutgers University, New Brunswick, New Jersey 08903.

† Department of Econometrics, Tilburg University, P. O. Box 90153, 5000 LE Tilburg, the Netherlands. The research of this author was supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.) through the Stichting Mathematisch Centrum.

‡ Department of Econometrics, Tilburg University, P. O. Box 90153, 5000 LE Tilburg, the Netherlands, and Mathematical Centre, Kruislaan 413, 1098 SJ Amsterdam, the Netherlands.

**2. Proof of Theorem 1.** To show Theorem 1, we first give a lemma. Let $E_1, \cdots, E_m$ be a collection of finite nonempty sets, which we order so that $|E_1|, \cdots, |E_h| \geqq 2$ and $|E_{h+1}| = \cdots = |E_m| = 1$, for some $h \leqq m$. We define a new collection as follows. Let

$$W := E_{h+1} \cup \cdots \cup E_m. \tag{4}$$

Let for each $i = 1, \cdots, h$, $X_i$ be a set of size $|E_i| - 2$, disjoint from $E_1 \cup \cdots \cup E_m$ and so that if $i \neq j$ then $X_i \cap X_j = \varnothing$. Let $X_1 \cup \cdots \cup X_h =: \{y_1, \cdots, y_q\}$. Then the *derived* collection of sets is formed by the following sets:

$$(E_1 \backslash W) \cup X_1, \cdots, (E_h \backslash W) \cup X_h, \{y_1\}, \cdots, \{y_q\}. \tag{5}$$

Furthermore, we define a collection $E_1, \cdots, E_m$ to have the $t$-SDR-*property* if any $t$ sets among $E_1, \cdots, E_m$ have a system of distinct representatives.

LEMMA. *For $t \geqq 3$, if $E_1, \cdots, E_m$ has the $t$-SDR-property, then the derived collection (5) has the $(t-2)$-SDR-property.*

*Proof.* Suppose (5) does not have the $(t-2)$-SDR-property. Then there exists a collection $\Pi$ of $p$ sets among (5) covering at most $p-1$ elements, for some $p \leqq t-2$. Assume we have chosen $p$ minimal. This immediately implies the following:

(6) (i) $|\cup \Pi| = p - 1$;
(ii) Each element in $\cup \Pi$ is covered by at least two sets in $\Pi$.

From (6)(ii) we directly have for any $i = 1, \cdots, h$ and $x \in X_i$:

$$\{x\} \in \Pi \Leftrightarrow (E_i \backslash W) \cup X_i \in \Pi. \tag{7}$$

Without loss of generality, all sets $(E_1 \backslash W) \cup X_1, \cdots, (E_h \backslash W) \cup X_h$ belong to $\Pi$ (as we can delete all sets $E_j$ from $E_1, \cdots, E_h$ for which $(E_j \backslash W) \cup X_j \notin \Pi$), and without loss of generality, $(E_1 \cup \cdots \cup E_h) \cap W = E_{h+1} \cup \cdots \cup E_m$.

Note the following:

$$q = |X_1 \cup \cdots \cup X_h| = \sum_{i=1}^{h} (|E_i| - 2), \qquad p = h + q, \tag{8}$$
$$\left| \bigcup_{i=1}^{h} (E_i \backslash W) \right| = |\cup \Pi| - q = (p-1) - q = h - 1.$$

So,

$$\left| \bigcup_{i=1}^{m} E_i \right| = \left| \bigcup_{i=1}^{h} (E_i \cap W) \right| + \left| \bigcup_{i=1}^{h} (E_i \backslash W) \right| = (m-h) + (h-1) = m - 1. \tag{9}$$

Moreover, by (6)(ii), $\sum_{i=1}^{h} |E_i \backslash W| \geqq 2 \cdot |\bigcup_{i=1}^{h} (E_i \backslash W)|$, and hence

$$\begin{aligned} m = h + \left| \bigcup_{i=1}^{h} (E_i \cap W) \right| &\leqq h + \sum_{i=1}^{h} |E_i \cap W| = h + \sum_{i=1}^{h} |E_i| - \sum_{i=1}^{h} |E_i \backslash W| \\ &\leqq h + \sum_{i=1}^{h} |E_i| - 2 \cdot \left| \bigcup_{i=1}^{h} (E_i \backslash W) \right| = h + 2h + \sum_{i=1}^{h} (|E_i| - 2) - 2(h-1) \\ &= h + 2h + q - 2(h-1) = h + q + 2 = p + 2 \leqq t. \end{aligned} \tag{10}$$

Inequalities (9) and (10) contradict the fact that $E_1, \cdots, E_m$ has the $t$-SDR-property. □

*Proof of Theorem* 1. We prove Theorem 1 by induction on $t$.

*Case* 1. $t = 1$. Then we have that each of $E_1, \cdots, E_m$ is nonempty, and hence $m \leqq \sum_{i=1}^{m} |E_i| \leqq kn$, by (1)(i).

*Case* 2. $t = 2$. Then we have that each of $E_1, \cdots, E_m$ is nonempty, and that no two of the singletons among $E_1, \cdots, E_m$ are the same. Without loss of generality, let $E_{h+1}, \cdots, E_m$ be the singletons among $E_1, \cdots, E_m$. Then $m - h \leqq n$, and

$$(11) \qquad m+h=2h+(m-h)\leqq \sum_{i=1}^{h} |E_i| + \sum_{i=h+1}^{m} |E_i| = \sum_{i=1}^{m} |E_i| \leqq kn$$

(by (1)(i)). Hence $2m = (m - h) + (m + h) \leqq (k + 1)n$, and (2) follows.

*Case* 3. $t \geqq 3$. Then consider the derived collection $E'_1, \cdots, E'_{m'}$ on $V' := \bigcup_{i=1}^{m'} E'_i$ as in (5). Note that $m' = h + q$ and $n' := |V'| = n - |W| + q$. Denote the right-hand side term in (2) by $\varphi(k, t)$.

As by the lemma above, $E'_1, \cdots, E'_{m'}$ has the $(t-2)$-SDR-property, and as trivially each element of $V'$ is in at most $k$ of the sets $E'_1, \cdots, E'_{m'}$ we have by induction that $m' \leqq \varphi(k, t-2)n'$. That is,

$$(12) \qquad h+q\leqq\varphi(k,t-2)(n-|W|+q).$$

Writing the terms in different order, we have

$$(13) \qquad \varphi(k,t-2)|W|+h-(\varphi(k,t-2)-1)q\leqq\varphi(k,t-2)n.$$

Moreover, as $E_1, \cdots, E_m$ cover any element at most $k$ times:

$$(14) \quad |W|+2h+q=|W|+2h+\sum_{i=1}^{h}(|E_i|-2)=|W|+\sum_{i=1}^{h}|E_i|=\sum_{i=1}^{m}|E_i|\leqq kn.$$

Hence,

$$(15) \qquad \begin{aligned} m&=h+|W| \\ &=\frac{1}{2\varphi(k,t-2)-1}(\varphi(k,t-2)|W|+h-(\varphi(k,t-2)-1)q) \\ &\quad+\frac{\varphi(k,t-2)-1}{2\varphi(k,t-2)-1}(|W|+2h+q) \\ &\leqq\frac{1}{2\varphi(k,t-2)-1}\varphi(k,t-2)n+\frac{\varphi(k,t-2)-1}{2\varphi(k,t-2)-1}kn \\ &=\frac{(k+1)\varphi(k,t-2)-k}{2\varphi(k,t-2)-1}n=\varphi(k,t)n. \end{aligned}$$

The last equality follows directly by substituting the corresponding right-hand side of (2). □

**3. Proof of Theorem 2.** To prove Theorem 2 we use a result of Erdös and Sachs [1]:

(16) For every $k$ and $\gamma$ there exists a $k$-regular graph of girth $\gamma$.

As a consequence of (16) we have the following:

(17) For every $k$, $s$, and $\gamma$ there exists a bipartite graph of girth at least $\gamma$, with color classes $U$ and $W$, say, such that each vertex in $U$ has degree $k$, and each vertex in $W$ has degree $s$.

(To see that (17) follows from (16), let $H$ be a $2ks$-regular graph of girth $\gamma$. Consider any Eulerian orientation of the edges of $H$ (i.e., one for which all indegrees and outdegrees equal $ks$). Split each vertex $v$ into $k + s$ vertices $v_1, \cdots, v_k, w_1, \cdots, w_s$ and divide the arcs entering $v$ equally over $v_1, \cdots, v_k$ and divide the arcs leaving $v$ equally over $w_1, \cdots, w_s$. Forgetting the orientations, we obtain a bipartite graph with the required properties.)

Now choose $k$, $t$. Let $r := \lfloor \frac{1}{2}t \rfloor$. Consider the tree $T$, with vertices $1, 2, \cdots, 1 + (k-1) + (k-1)^2 + \cdots + (k-1)^{r-1}$, so that for $i < j$, vertices $i$ and $j$ are connected by an edge, if and only if $(k-1)i \leqq j \leqq (k-1)i + (k-2)$. So each vertex has degree $k$, except for vertex 1, which has degree $k - 1$, and for the vertices $1 + (k-1) + \cdots + (k-1)^{r-2} + 1, \cdots, 1 + (k-1) + \cdots + (k-1)^{r-1}$, which have degree one.

First let $t$ be even. Let $G$ be a $(k-1)^r$-regular graph of girth $t + 1$ (cf. (16)). Let $G$ have $p$ vertices: $v_1, \cdots, v_p$. Consider $p$ copies $T_1, \cdots, T_p$ of $T$ (denoting the copy of vertex $i$ in $T_j$ by $i_j$). For each $j = 1, \cdots, p$, partition the set of $(k-1)^r$ edges of $G$ incident to $v_j$ (arbitrarily) into $(k-1)^{r-1}$ classes of size $k - 1$, and connect them to the $(k-1)^{r-1}$ vertices $i_j$ in $T_j$ of degree one. So the final graph $H = (W, F)$ has all degrees equal to $k$, except for the vertices $1_1, \cdots, 1_p$, which have degree $k - 1$. Let $E_1, \cdots, E_m$ be the collection $F \cup \{\{1_1\}, \cdots, \{1_p\}\}$. This collection clearly satisfies (1)(i), and direct counting shows equality in (2)(ii). To see that the collection satisfies (1)(ii), let $E_1, \cdots, E_s$ form a subcollection with $|E_1 \cup \cdots \cup E_s| < s$ and $s$ as small as possible. Suppose $s \leqq t$. As $E_1, \cdots, E_s$ must form a connected hypergraph, it contains at most one singleton (since any path between $1_i$ and $1_j$ in $H$ contains at least $t - 1$ edges). So assume $E_2, \cdots, E_s$ are edges of $H$. Then they do not contain any circuit (as each $T_i$ is a tree and as $G$ has girth $t + 1 > s$). So $|E_2 \cup \cdots \cup E_s| \geqq s$, a contradiction.

Next let $t$ be odd. Let $G$ be a bipartite graph, of girth at least $t + 1$, so that in one color class $U$ each vertex has degree $(k-1)^r$ and in the other color class $W$ each vertex has degree $k$. Let $U =: \{u_1, \cdots, u_p\}$. Consider again $p$ copies $T_1, \cdots, T_p$ of $T$, as above. For $j = 1, \cdots, p$ partition the set of $(k-1)^r$ edges of $G$ incident to $u_j$ (arbitrarily) into $(k-1)^{r-1}$ classes of size $k - 1$, and connect them to the $(k-1)^{r-1}$ vertices $i_j$ in $T_j$ of degree one. Again, the final graph $H = (W, F)$ has all degrees equal to $k$, except for the vertices $1_1, \cdots, 1_p$ that have degree $k - 1$. Let $E_1, \cdots, E_m$ be the collection $F \cup \{\{1_1\}, \cdots, \{1_p\}\}$. Similarly, as above, we show that this collection satisfies (1) and has equality in (2)(i).

**4. Application to the worst-case ratio of heuristics.** The problem of finding a largest collection of pairwise disjoint sets among a given collection $X_1, \cdots, X_q$ of $k$-sets is NP-complete, for any $k \geqq 3$. Call any collection of pairwise disjoint sets a *packing*.

For any fixed $s$, we can apply the following heuristic algorithm $H_s$. Start with the empty packing. If we have found a packing $Y_1, \cdots, Y_n$ from $X_1, \cdots, X_q$, we could select $p \leqq s$ sets among $Y_1, \cdots, Y_n$, and replace them by $p + 1$ sets from $X_1, \cdots, X_q$, so that the arising collection is a packing with $n + 1$ sets. Repeating this, the algorithm terminates with a collection $Y_1, \cdots, Y_n$ so that

(18) For each $p \leqq s$, the union of any $p + 1$ pairwise disjoint sets among $X_1, \cdots, X_q$ intersects at least $p + 1$ sets among $Y_1, \cdots, Y_n$.

This defines heuristic $H_s$, which is, for any fixed $s$, a polynomial-time algorithm—however it clearly need not lead to a largest packing. We might ask how far the packing found with $H_s$ is from the largest packing.

To this end, consider a largest packing $Z_1, \cdots, Z_m$ from $X_1, \cdots, X_q$. We claim that $m/n$ satisfies the bounds given in (2), taking $t := s + 1$, and that these bounds are best possible. That is, the "worst-case ratio" of the heuristic is given in (2).

Indeed, let

(19) $V := \{Y_1, \cdots, Y_n\}$ and $E_i := \{Y_j \mid Y_j \cap Z_i \neq 0\}$ for $i = 1, \cdots, m$.

Then by (18), $E_1, \cdots, E_m$ satisfy (1), and hence we obtain the bounds given in (2).

In turn, it is not difficult to see that for any collection $E_1, \cdots, E_m$ of sets of size at most $k$, containing any point at most $k$ times, we can assume they are of form (19) for certain packings $Y_1, \cdots, Y_n$ and $Z_1, \cdots, Z_m$ of $k$-sets. Thus starting with $E_1, \cdots, E_m$ as described in § 3 above, making these $Y_1, \cdots, Y_n$, $Z_1, \cdots, Z_m$, and taking $\{X_1, \cdots, X_q\} := \{Y_1, \cdots, Y_n, Z_1, \cdots, Z_m\}$, we obtain a system of sets attaining the worst-case ratio. (That is because we may assume that $H_s$ selects the sets $Y_1, \cdots, Y_n$ in the first $n$ iterations.)

Note that we may assume even that the sets $Y_1, \cdots, Y_n$, $Z_1, \cdots, Z_m$ form the collection of all cliques of size $k$ in a graph. Hence, we cannot obtain a better worst-case ratio by restricting the collections of sets to collections of $k$-cliques.

## REFERENCE

[1] P. Erdős and H. Sachs, *Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl*, Wiss. Z. Univ. Halle, Math.-Nat., 12 (1963), pp. 251–258.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 73-90, February 1989

# ON SOLVING RATIONAL CONGRUENCES*

GERHARD JAESCHKE†

**Abstract.** In this paper some extensions of the Farey fraction method as presented in [*Methods and Applications of Error-Free Computation*, Springer-Verlag, Berlin, New York, 1984] are discussed for determining solutions of systems of the form $x \equiv ay \bmod m$ in unknown integers $x$, $y$ where $x$, $y$ are restricted by $0 \leqq x \leqq N$, $1 \leqq y \leqq N$, $\gcd(y, m) = 1$. Since $x \equiv ay \bmod m$ can be used as definition of $x/y \equiv a \bmod m$ such a system is spoken of as a "rational congruence" with bounded variables $x$, $y$. Interval conditions are analyzed for $x/y$ to enlarge the set of rational congruences that can be handled by procedures of Euclidean type (gcd computations). Several approaches are proposed for solving rational congruences with one or more moduli.

**Key words.** congruences, residue classes, continued fractions, Euclidean processes

**AMS(MOS) subject classifications.** primary 10A10, 10A32; secondary 15A06

**0. Introduction.** One important alternative for error-free computations with rational numbers (i.e., without using floating point numbers for intermediate calculations) is modular arithmetic where residue number systems are used for addition, subtraction, and multiplication of rational numbers.

Residue number systems allow arithmetic on large integers [Kn81] and especially long integer multiplication can be based efficiently on modular methods [Sc66], [SS71]. They can also be applied to ill-conditioned systems of linear equations where the coefficients are rationals with relatively small denominators [GK84].

The main idea of modular arithmetic is to work with "small" integers (as representatives of residue classes) instead of working with rational numbers or "large" integers. Therefore, each modular calculation requires the solution of three subproblems: (1) forward mapping (from rational numbers to residues); (2) computing residual results; and (3) backward mapping (from residues to rational numbers).

In the case of integer input and output data the principal problems have been solved (by use of the Chinese Remainder Theorem and, e.g., Mixed Radix Representation of Integers), but in the case of rational input and output data only the first two of the above-mentioned subproblems are solved in a satisfactory way. The third step of a modular computation with rational numbers is crucial and further research is needed. So we are motivated to present some new results on the solution of rational congruences in this note that go beyond the basic results contained in the book *Methods and Applications of Error-Free Computation* by Gregory and Krishnamurthy [GK84].

It should be pointed out that solving systems of linear equations with rational coefficients leads to rational congruences when the coefficients are directly modular coded and not transformed into integers by multiplications. This can be seen as follows.

Let $A$ be a $n$ by $n$ matrix and let $X$, $b$ be column vectors. Assume further that

$$AX = b$$

where

$$A = \left(\frac{a_{ij}}{b_{ij}}\right), \quad X = \left(\frac{x_j}{y_j}\right), \quad b = \left(\frac{c_i}{d_i}\right),$$

and $a_{ij}$, $b_{ij}$, $c_i$, $d_i$, $x_j$, $y_j$ are integers with $b_{ij}$, $y_j$, $d_i \neq 0$.

* Received by the editors November 2, 1987; accepted for publication (in revised form) July 1, 1988.

† IBM Heidelberg Science Center, 6900 Heidelberg, Federal Republic of Germany.

Now if $m_1, \cdots, m_r$ are pairwise relatively prime moduli and $\gcd(b_{ij}, m_k) = \gcd(d_i, m_k) = 1$ for all $i, j, k$, then from the system of equations we obtain the following for $i = 1, \cdots, n$ and $k = 1, \cdots, r$:

$$\sum_j \alpha_{ij}^{(k)} z_j \equiv \beta_i^{(k)} \bmod m_k$$

where

$$\alpha_{ij}^{(k)} \equiv a_{ij} b_{ij}^{-1} \bmod m_k, \quad \beta_i^{(k)} \equiv c_i d_i^{-1} \bmod m_k, \quad z_j = \frac{x_j}{y_j}, \quad \gcd(y_j, m_k) = 1.$$

Under the assumption that for each $k$ the matrix $(\alpha_{ij}^{(k)})$ is invertible mod $m_k$ the above system of congruences is equivalent to the system

$$z_j \equiv \gamma_j^{(k)} \bmod m_k, \qquad j = 1, \cdots, n, \quad k = 1, \cdots, r$$

with certain integers $\gamma_j^{(k)}$ in $[0, m_k - 1]$, or equivalent to (by Chinese remainder conversion)

$$z_j \equiv \delta_j \bmod \prod_{k=1}^{r} m_k$$

for $j = 1, \cdots, n$, with $0 \leqq \delta_j \leqq \prod_{k=1}^{r} m_k - 1$. For each $j$ the last relationship is a "rational congruence" in our sense, which has a unique solution when $r$ is large enough.

In § 1 we briefly discuss the solution method of Gregory and Krishnamurthy, then try to apply smaller moduli and present a new method for solving two-modulus systems (of rational congruences). In § 2 we analyze interval conditions to enlarge the range of rational congruences that can be solved, using smaller moduli than in Gregory and Krishnamurthy's method. Here we present a basic theorem, an algorithm, and two application methods for solving such interval conditioned rational congruences. In addition, a procedure is presented for solving multimodulus congruences with interval conditions. Finally, in § 3 some remarks are made with respect to the distribution of so-called $m$-bounded fractions in generalized residue classes.

**0.1. Notational remarks.**

$\gcd(a, b)$ = greatest common divisor of $a$ and $b$,
$[x]$ = integer part of the real number $x$,
$\mathbb{Z}$ = set of integers,
$\mathbb{N}$ = set of positive integers,
$m$ = product of the moduli $m_1, \cdots, m_r$,
$\tilde{Q}$ = set of rational numbers with denominator relatively prime to $m$,
$\tilde{Q}_k$ = generalized residue class mod $m$, for $k$ in the range $0 \leqq k \leqq m - 1$,
$F_N$ = set of order-$N$ Farey fractions (defined in § 1),
$L_m$ = set of $m$-bounded fractions (defined in § 3),
$L_{dm}$ = $m$-bounded residue class $d$ mod $m$ (defined in § 3).

**1. Solution of rational congruences.** For each positive integer $m$ we define as in [GK84] so-called "generalized residue classes" mod $m$: $\tilde{Q}_0, \cdots, \tilde{Q}_{m-1}$ if we put

$$\tilde{Q}_k = \left\{ \frac{a}{b} \,\middle|\, \frac{a}{b} \in \tilde{Q} \wedge a \equiv k \cdot b \bmod m \right\}$$

with $\tilde{Q}$ as defined in § 0.1. The main difference between the usual residue classes $Q_k$ (equal to a set of all integers $\equiv k \bmod m$) and the generalized residue classes $\tilde{Q}_k$ is that for $k$ not equal to zero there exists exactly one element of $Q_k$ between two multiples of

$m$ but in the same interval there are infinitely many rational numbers belonging to $\tilde{Q}_k$. For example, the rational numbers in the interval $[0, m]$ belonging to $\tilde{Q}_k$ for $1 \leqq k < m$ include the infinitely many numbers

$$k - \frac{m}{b} \cdot \left[\frac{bk}{m}\right], \quad b \in \mathbb{N}, \quad \gcd(b, m) = 1.$$

This situation requires the choice of convenient subsets $F_N$ of $\tilde{Q}$ such that each $F_N \cap \tilde{Q}_k$ contains at most one element that can be identified by an efficient algorithm if $F_N \cap \tilde{Q}_k$ is not empty. So, in [GK84] $F_N$ is taken to be the set of order-$N$ "Farey fractions," that is,

$$F_N = \left\{ \frac{a}{b} \,\middle|\, \frac{a}{b} \in \tilde{Q} \wedge \gcd(a,b) = 1 \wedge 0 < |a|, |b| \leqq N \right\}.$$

Now, in [GK84] it is shown that given a positive integer $m$ and integer $d$ with $0 < d < m$ there exists at most one irreducible fraction $a/b$ such that

$$\frac{a}{b} \equiv d \bmod m, \tag{1.1}$$

$$|a|, |b| \leqq \sqrt{(m-1)/2} \tag{1.2}$$

hold. If there is a fraction $a/b$ for which (1.1), (1.2) are satisfied, then this fraction is obtainable by a Euclidean process as follows.

Define a sequence of pairs $(a_i, b_i)$ by putting

$$a_{-1} = m, \qquad b_{-1} = 0, \tag{1.3}$$

$$a_0 = d, \qquad b_0 = 1, \tag{1.4}$$

$$a_i = a_{i-2} - a_{i-1} \cdot \left[\frac{a_{i-2}}{a_{i-1}}\right], \tag{1.5}$$

$$b_i = b_{i-2} - b_{i-1} \cdot \left[\frac{a_{i-2}}{a_{i-1}}\right]. \tag{1.6}$$

Here

$$\begin{pmatrix} a_{-1} & b_{-1} \\ a_0 & b_0 \end{pmatrix}$$

is called the "seed matrix" of the process. The index $i$ runs from 1 through $n + 1$ as long as $a_{i-1}$ is unequal zero.

Assume that $a$, $b$ exist satisfying (1.1), (1.2). Then, by a theorem of Kornerup (see [GK84, Thm. 6.39]), there exists an index $i$ such that

$$|a_i| = |a|, \qquad |b_i| = |b|$$

where $a_i$, $b_i$ are defined as above. This means that the above-introduced order-$N$ Farey fractions can always be recovered if $N$ is chosen such that in (1.1) $m$ is greater than $2 \cdot N^2$. Conversely, if we know that $a$ and $b$ are integers between 0 and $N$ we should use a modulus $m$ greater than $2 \cdot N^2$.

*Example*. Let us assume that $0 \leqq a$, $b \leqq 20$ and $a/b \equiv 76 \bmod 829$. Find $a$ and $b$. We have the following schema:

| | | |
|---|---|---|
| | 829 | 0 |
| | 76 | 1 |
| 10 | 69 | −10 |
| 1 | 7 | 11 |
| 9 | 6 | −109 |
| 1 | 1 | 120 |

from which we derive $a/b = 7/11$. The left-hand side integers 10, 1, 9, 1 are the quotients $[a_{i-2}/a_{i-1}]$ for $i = 1, \cdots, 4$.

*Remark.* All quotients $a_i/b_i$ belong to $\tilde{Q}_d$.

The solution of the rational congruence (1.1) by the Farey fraction method requires a modulus $m$ greater than $1 + 2\cdot(\max\{|a|, |b|\})^2$. The question arises whether there exist smaller moduli $m$ such that $a/b$ can be recovered by Euclidean processes. To analyze this question we introduce two useful sets $V_{dm}$ and $R_{ab}$ as follows.

Define a Euclidean process as in (1.3)–(1.6), and let

$$V_{dm} = \{(a_i, b_i) \mid 1 \leqq i \leqq n\}, \tag{1.7}$$

$$R_{ab} = \{t \mid t > \max\{|a|, |b|\} \wedge (a, b) \in V_{(a/b) \bmod t, t}\} \tag{1.8}$$

where $\gcd(a, b) = 1$. This means that $V_{dm}$ is the set of all pairs $a_i$, $b_i$ occurring in a row of the Euclidean schema produced by the seed matrix

$$\begin{pmatrix} m & 0 \\ d & 1 \end{pmatrix}$$

and that $R_{ab}$ is the set of moduli $t$ for which a given pair $(a, b)$ occurs in a row of the Euclidean schema corresponding to the seed matrix

$$\begin{pmatrix} t & 0 \\ d & 1 \end{pmatrix}$$

where $a \equiv bd \bmod t$.

*Example.* From the Euclidean schema given above we derive

$$V_{76,829} = \{(69, -10), (7, 11), (6, -109), (1, 120)\} \quad \text{and} \quad 829 \in R_{7,11}. -.$$

From [GK84] we have the result that $t \in R_{ab}$ for all $t \geqq 1 + 2\cdot\max\{a, b\}^2$ where $a, b \in \mathbb{N}$ are given and $\gcd(t, b) = 1$. But it is easy to see that $R_{ab}$ contains smaller elements $t$. So we prove the following theorem.

THEOREM 1. *For* $0 < a < b$ *and* $\gcd(a, b) = 1$ *we have* $a + k\cdot b \in R_{ab}$ *for all* $k \geqq a + 1$.

*Proof.* Since $a/b \equiv -k \bmod (a + kb)$ the Euclidean schema is as follows:

| | | |
|---|---|---|
| | $a + kb$ | 0 |
| | $a + kb - k$ | 1 |
| 1 | $k$ | −1 |
| $b - 1$ | $a$ | $b$ |
| $\cdots$ | $\cdots$ | $\cdots$ |

(since $[(a + kb - k)/k] = b - 1$ because of $k \geqq a + 1$).

COROLLARY. *Under the same assumptions as in Theorem* 1 *we have* $ab + a + b \in R_{ab}$.

THEOREM 2. $a + k \cdot b \notin R_{ab}$ *for all* $k \leqq a < b$.

*Proof*. The Euclidean schema under consideration here is

| | | |
|---|---|---|
| | $a + kb$ | $0$ |
| | $a + kb - k$ | $1$ |
| $1$ | $k$ | $-1$ |
| $\dots$ | $<a$ | $\dots$ |

.

If $k < a$ then all $a_i < a$, so that $a$ does not occur in the first column. If $k = a$ then we have by view of $b_1 = -1$ the result $b \neq b_1$ but $a_1 = a$, so that the pair $(a, b)$ cannot occur in any row of the above schema. □

As consequence of Theorems 1 and 2 we find that $t = ab + a + b$ is the smallest element in $R_{ab}$ of the form $a + k \cdot b$. We conjecture that $ab + a + b$ is the minimum element of $R_{ab}$ at all but this is unsettled up to now. Also, no counterexample has been found.

CONJECTURE 1. $ab + a + b = \min R_{ab}$.

Our next (more important) concern with respect to $R_{ab}$ is to determine the smallest $\rho \in R_{ab}$ with the property that all $t \geqq \rho$ with $\gcd(t, b) = 1$ belong to $R_{ab}$. So we define

$$\rho_{ab} = \min \{t \mid (\forall x)(x \geqq t \wedge \gcd(x, b) = 1 \rightarrow x \in R_{ab})\} \tag{1.9}$$

and find Theorem 3.

THEOREM 3. $\rho_{ab} \geqq 2ab - a + 1$.

*Proof*. For $t = 2ab - a$ we have the following schema:

| | | |
|---|---|---|
| | $2ab - a$ | $0$ |
| | $2a$ | $1$ |
| $b - 1$ | $a$ | $1 - b$ |
| $\dots$ | $\dots$ | $\dots$ |

;

hence $(a, b) \notin V_{2a,t}$ and $2ab - a \notin R_{ab}$.

*Remark*. The inequality in Theorem 3 cannot be strengthened since, for example, when $a = 10$ and $b = 13$ then we have $t \in R_{10,13}$ for all $t > 250$ with $t \not\equiv 0 \bmod 13$.

From the proof of Theorem 6.39 in [GK84] we can see that for $a/bt < 1/2b^2$ we have $t \in R_{ab}$ from which it follows that $\rho_{ab} \leqq 2ab + 1$. Together with Theorem 3 we therefore have

$$2ab - a + 1 \leqq \rho_{ab} \leqq 2ab + 1. \tag{1.10}$$

As we will show (1.10) suffices for our applications, and in no case did we find $\rho_{ab} > 2ab - a + 1$. So we have Conjecture 2.

CONJECTURE 2. $\rho_{ab} = 2ab - a + 1$.

This could not be proved up until now.

The previous results suggest the following solution method (called "intersection method") for solving two-modulus systems without using the Chinese Remainder Theorem. When we must solve the system

$$\frac{a}{b} \equiv d \bmod m \wedge \frac{a}{b} \equiv e \bmod n \tag{1.11}$$

with $1 + 2 \cdot \max\{|a|, |b|\} \leqq mn$, and coprime $m, n$, then principally we could apply Chinese remainder conversion, which yields only one congruence:

$$\frac{a}{b} \equiv f \bmod mn \tag{1.12}$$

and recover $a/b$ as the Farey fraction of this system. Instead of using the product modulus $m, n$ to solve (1.12), we apply the Farey fraction method to the two congruences in (1.11) separately and determine a common row of the obtained Euclidean schemata. For this purpose we must choose the moduli $m, n$ appropriately in order to guarantee the existence of such a common row. Now let us assume that we know $ab < \beta$. Then we choose two primes $m, n > 2\beta$ and determine $d = (a/b) \bmod m$, $e = (a/b) \bmod n$. From (1.10) it follows that $m, n \in R_{ab}$ and therefore $(a, b)$ can be recovered as the only element of $V_{dm} \cap V_{en}$. But $V_{dm}$ and $V_{en}$ are obtained by Euclidean processes in the sense of [GK84].

*Example* 1. Let $ab < 40$ and $a/b \equiv 36 \bmod 83$ and $a/b \equiv 3 \bmod 89$. Then $V_{36,83} \cap V_{3,89} = \{(1, 30)\}$; hence $a/b = 1/30$.

*Example* 2. Assume that we know $b < 200$ and $a/b \equiv 0 \bmod 4$ and $0.55 < a/b < 0.56$. Since $4/7$ is a continued fraction convergent of 0.56, we put $u/v = 1 - 7a/4b$. Then we have $0 < u/v < 1/25$ and $v < 200$; hence $u < 8$ and $uv < 1600$. So we try $m = 1609$ and $n = 1601$ as moduli and compute $(a/b) \bmod mn$. Let us obtain

$$\frac{a}{b} \equiv 259 \bmod 1609 \wedge \frac{a}{b} \equiv 996 \bmod 1601.$$

This implies

$$\frac{u}{v} \equiv 1559 \bmod 1609 \wedge \frac{u}{v} \equiv 1460 \bmod 1601.$$

We find $4/193 \in V_{1460,1601} \cap V_{1559,1609}$; hence, $u/v = 4/193$ and

$$a/b = 4/7 \cdot (1 - u/v) = 108/193.$$

*Remark*. To guarantee that $u/v$ can be recovered we should take $m, n > 3200$; but in many cases smaller moduli are sufficient.

**2. Rational congruences with interval conditions.** The range of fractions that are recoverable can be enlarged considerably by prescribing certain interval conditions that the fraction must satisfy. Thus, condition (1.2) can be weakened to

$$0 < a, b < m. \tag{2.1}$$

The additional interval requirement is

$$\frac{h}{m} < \frac{a}{b} < \frac{h+1}{m} \tag{2.2}$$

for an integer $h$.

Then we prove the following:

(1) There exists at most one irreducible fraction $a/b$ that satisfies (1.1), (2.1), and (2.2).

(2) If such a fraction exists, then it can be rediscovered by two Euclidean processes as a common continued fraction convergent of two rational numbers depending on $d$, $h$, $m$.

We begin with Theorem 4.

THEOREM 4. *Let $I$ be an interval of length less than $1/m$ and let $d$ be an integer with $1 \leqq d < m$. Then there is at most one rational number $a/b \in I$ with $1 \leqq a, b < m \wedge \gcd(a,b) = 1 \wedge a/b \equiv d \bmod m$. If a rational number $a/b$ satisfies these conditions, then $a/b \leqq d$.*

*Proof.* Let $1 \leqq a, b, a', b' < m$ with $a/b, a'/b' \in I$, and $a/b \equiv a'/b' \bmod m$ and $\gcd(a,b) = \gcd(a',b') = 1$. Without loss of generality we assume $a/b \leqq a'/b'$; hence $0 \leqq a'/b' - a/b < 1/m$ and therefore $0 \leqq a'b - ab' < bb'/m < m$.

On the other hand, $m$ divides $a'b - ab'$; hence $a'b - ab' = 0$ and $a/b = a'/b'$. From $\gcd(a,b) = \gcd(a',b') = 1$ we even conclude $a = a'$ and $b = b'$. To prove the second part of the assertion let $\lambda \in \mathbb{Z}$ such that $a = bd + \lambda m$. Then from $a < m$ and $bd > 0$ it follows that $\lambda$ cannot be positive and so we have $a/b \leqq d$.

In the following we can restrict ourselves to the case $h < m$ since otherwise we have $[h/m] = k \geqq 1$ and the inequality $h/m < a/b < (h+1)/m$ is equivalent to $h'/m < a'/b < (h'+1)/m$ where $h' = h - km$ and $a' = a - kb$.

THEOREM 5. *Under the conditions $a/b \equiv d \bmod m$ and $0 < a, b < m$ the inequalities*

$$\frac{h}{m} < \frac{a}{b} < \frac{h+1}{m} \tag{2.3}$$

*and*

$$\frac{\alpha}{m^2} < \frac{g}{b} < \frac{\alpha+1}{m^2} \quad \textit{for some integer } g \tag{2.4}$$

*are equivalent, where $1 \leqq g, b, h < m \wedge \gcd(b,g) = 1 \wedge \alpha = dm - h - 1$.*

*Proof.* $0 < a < m$ and $a/b \equiv d \bmod m$ imply $a = bd - m \cdot [bd/m]$. Hence, (2.3) is equivalent to

$$\frac{bh}{m} < bd - m \cdot \left[\frac{bd}{m}\right] < \frac{b(h+1)}{m},$$

and therefore to

$$\frac{\alpha}{m^2} < \frac{1}{b} \cdot \left[\frac{bd}{m}\right] < \frac{\alpha+1}{m^2} \tag{2.5}$$

with $\alpha = dm - h - 1$. This proves (2.3) $\rightarrow$ (2.4). Conversely, if (2.4) holds, then we obtain

$$\frac{bh}{m^2} < \frac{bd}{m} - g < \frac{b(h+1)}{m^2};$$

hence, by the assumptions of the theorem $0 < bd/m - g < 1$ and $g = [bd/m]$. But this gives (2.5); hence we have (2.3).

*Remark.* In (2.4) $g$, $b$ are determined uniquely since if also $\alpha/m^2 < g'/b' < (\alpha+1)/m^2$ with $g', b' < m$, then we have $|g/b - g'/b'| < 1/m^2$ contradicting

$$\left|\frac{gb' - bg'}{bb'}\right| \geqq \frac{1}{bb'} > \frac{1}{m^2} \quad \text{for } \frac{g}{b} \neq \frac{g'}{b'}.$$

Finally, we turn over to the solution of inequality (2.4) in integers $b$, $g$ between 1 and $m - 1$. We generate successively irreducible fractions $a_i/b_i$ and $c_i/d_i$ with decreasing denominators such that

$$\frac{a_0}{b_0}=\frac{\alpha}{m^2} \quad\text{and}\quad \frac{c_0}{d_0}=\frac{\alpha+1}{m^2}$$

as well as

$$a_{i+1}b_i-a_ib_{i+1}=1, \qquad c_id_{i+1}-c_{i+1}d_i=1.$$

The first sequence is increasing; the second one is decreasing. Then if $a_i/b_i = c_j/d_j$ for a pair $i, j$, then $g = a_i$, $b = b_i$ is the desired solution. It remains to show that there always exists such a pair $i, j$. To demonstrate this we need the following simple lemma.

LEMMA. *If* $a/b < x/y < c/d$, $bc - ad = 1$ *and* $a, b, c, d, x, y > 0$, *then* $y \geqq b + d$.

*Proof.* Since $a, b, c, d, x, y$ are supposed to be integers, we conclude from the assumption $bx - ay \geqq 1$ and $cy - dx \geqq 1$ the inequalities $x/y - a/b \geqq 1/by$ and $c/d - x/y \geqq 1/dy$. By adding these inequalities we obtain

$$\frac{c}{d}-\frac{a}{b}\geqq\frac{1}{y}\cdot\left(\frac{1}{b}+\frac{1}{d}\right),$$

which gives $bc - ad \geqq (b + d)/y$; hence $y \geqq b + d$. □

Because of the uniqueness in (2.4) every fraction between $g/b$ and $\alpha/m^2$, on the one hand, and between $g/b$ and $(\alpha + 1)/m^2$, on the other hand, has a denominator greater than $m$, hence greater than $b$, from which we conclude by Theorem 21 of [Pe58] that $g/b$ is a continued fraction convergent (main or not main) of $\alpha/m^2$ as well as of $(\alpha + 1)/m^2$. But the above-generated sequences contain, by means of the lemma and Theorem 20 of [Pe58], all continued fraction convergents of $\alpha/m^2$ greater than $\alpha/m^2$, and of $(\alpha + 1)/m^2$ smaller than $(\alpha + 1)/m^2$; thus the desired pair $i, j$ with $a_i/b_i = c_j/d_j = g/b$ exists.

*Remark.* The numerator of $a/b$ can be obtained from $a \equiv bd \bmod m$.

*Example.* Solve $a/b \equiv 17 \bmod 97 \wedge 42/97 < a/b < 43/97$. We find as in inequality (2.3) that

$$\frac{1606}{9409}<\frac{g}{b}<\frac{1607}{9409}.$$

*Ascending fractions.*

$$\frac{1606}{9409}<\frac{1323}{7751}<\frac{1040}{6093}<\frac{757}{4435}<\frac{474}{2777}<\frac{191}{1119}<\frac{99}{580}<\frac{7}{41}<\cdots.$$

*Descending fractions.*

$$\frac{1607}{9409}>\frac{469}{2746}>\frac{269}{1574}>\frac{69}{404}>\frac{7}{41}>\cdots.$$

So we have $g = 7$, $b = 41$, and $a \equiv 17\cdot 41 \bmod 97 = 18$; this means $a/b = 18/41$ is the desired fraction.

It is clear that the restriction (2.2) is a very strong one since the inclusion in the interval $[h/m, (h + 1)/m]$ requires a very good estimation of the solution $a/b$.

Therefore, we present in the following several classes of problems that can be solved by transforming them to the basic case of this section.

**2.1. Reciprocal approach.** We showed above that the system

(2.6) $$0<a,b<m,$$

(2.7) $$\frac{a}{b} \equiv d \bmod m, \qquad \gcd(d,m)=1,$$

(2.8) $$\alpha < \frac{a}{b} < \beta, \qquad \alpha > 0,$$

(2.9) $$\beta - \alpha < \frac{1}{m}$$

can be solved by two Euclidean processes. Now, we demonstrate Theorem 6.

THEOREM 6. *When we replace the condition* (2.9) *by*

(2.10) $$\frac{1}{\alpha} - \frac{1}{\beta} < \frac{1}{m}$$

*then the system* (2.6), (2.7), (2.8), (2.10) *can also be solved by the basic method described above.*

*Proof.* The system (2.6), (2.7), (2.8) is equivalent to the system

(2.11) $$0 < b, a < m,$$

(2.12) $$\frac{b}{a} \equiv d^{-1} \bmod m, \qquad \gcd(d,m)=1,$$

(2.13) $$\frac{1}{\beta} < \frac{b}{a} < \frac{1}{\alpha}.$$

*Example.* Let the system to be solved be

(EX1) $$0 < a, b < 4200013,$$

(EX2) $$\frac{a}{b} \equiv 3355579 \bmod 4200013,$$

(EX3) $$7215 < \frac{a}{b} < 7216.$$

Since $1/7215 - 1/7216 = 1/7215 \cdot 7216 < 1/4200013$ we can transform the system by interchanging $a$ and $b$:

(EX4) $$0 < b, a < 4200013,$$

(EX5) $$\frac{b}{a} \equiv 610942 \bmod 4200013,$$

(EX6) $$\frac{1}{7216} < \frac{b}{a} < \frac{1}{7215}.$$

By rewriting (EX6) such that it takes the form of condition (2.2), we have

(EX7) $$\frac{582}{4200013} < \frac{b}{a} < \frac{583}{4200013}.$$

We solve the system (EX4), (EX5), (EX7) by the basic method and obtain $b/a = 577/4163103$ and therefore $a/b = 7215(48/577)$.

It should be noted that this "reciprocal approach" needs an extra Euclidean process since the multiplicative inverse of $d$ mod $m$ in $a/b \equiv d \bmod m$ must be computed.

**2.2. Prime square approach.** The so-called prime square method for solving rational congruences is based on the following theorem.

THEOREM 7. *Let p, q be primes. Then the following system can be transformed to standard form* (2.6)–(2.9) *and thus can be solved by two Euclidean processes*:

$$0<a,b<c, \tag{2.14}$$

$$\frac{a}{b}\equiv 0 \bmod p, \quad \frac{a}{b}\equiv e \bmod p^2, \quad \frac{a}{b}\equiv d \bmod q, \tag{2.15}$$

$$\alpha<\frac{a}{b}<\beta, \tag{2.16}$$

$$\beta-\alpha<\frac{1}{q}<\frac{2p}{3c}, \tag{2.17}$$

$$p+\frac{a}{b}\not\equiv 0 \bmod p^2, q. \tag{2.18}$$

*Proof*. We apply the transformation $x \to f(x) = p/(p+x)$ to the system (2.14)–(2.18) to obtain a solvable system of standard type (see (2.6)–(2.9)). By (2.15) the integer $a$ is divisible by $p$. Let $a = p\lambda$; then we have

$$\frac{a'}{b'}=f\left(\frac{a}{b}\right)=\left(1+\frac{\lambda}{b}\right)^{-1}. \tag{2.19}$$

If $e' = e/p$ from (2.15) we obtain $\lambda/b \equiv e' \bmod p$; hence, by (2.19)

$$\frac{a'}{b'}\equiv(1+e')^{-1} \bmod p. \tag{2.20}$$

We put $m = pq$ and solve the linear congruence system

$$d'\equiv 0 \bmod p \wedge d'\equiv d \bmod q.$$

For $d'' = d'/p$ we have $\lambda/b \equiv d'' \bmod q$; hence

$$\frac{a'}{b'}\equiv(1+d'')^{-1} \bmod q. \tag{2.21}$$

The inverses in (2.20), (2.21) exist by (2.18). Formulae (2.20) and (2.21) yield

$$\frac{a'}{b'}\equiv m' \bmod m \tag{2.22}$$

if $m'$ is the solution of the system

$$m'\equiv(1+e')^{-1} \bmod p, \qquad m'\equiv(1+d'')^{-1} \bmod q$$

with $m = pq$. Since $a'/b' = b/(b+\lambda)$ and $b+\lambda = b + a/p < 3c/2$ we have $0 < a'$, $b' < c'$ with $c' = 3c/2$.

Formula (2.17) yields $c' < m$; hence

$$0<a',b'<m. \tag{2.23}$$

Because of the monotonicity of the function $f$, (2.16) is transformed into

$$\alpha'<\frac{a'}{b'}<\beta' \quad \text{with}\ \alpha'=\frac{p}{p+\beta}, \quad \beta'=\frac{p}{p+\alpha}. \tag{2.24}$$

It remains to show that $\beta' - \alpha' < 1/m$. But from $\beta - \alpha < 1/q$ it follows that

$$\beta - \alpha < \frac{1}{q} \cdot \left(1 + \frac{\alpha}{p}\right)\left(1 + \frac{\beta}{p}\right) \quad \text{and} \quad \beta' - \alpha' = p \cdot \frac{\beta - \alpha}{(p+\alpha)(p+\beta)} < \frac{1}{pq} = \frac{1}{m}.$$

So we have

(2.25) $$\beta' - \alpha' < \frac{1}{m}$$

and the system (2.20)–(2.23) is solvable by Euclidean processes. This completes our proof. □

While the range of applicability of the prime square method is much greater than that of our basic method, it requires some amount of additional time to find the appropriate moduli $p$, $q$. But the problem of determining appropriate moduli arises in any case where rational output is expected when modular methods are applied. We shall give an example of application of the prime square method after having proposed an algorithm for this purpose; then we shall exhibit some difficulties of the method, and finally, the method will be extended for the case where the second condition of the theorem does not hold for small $p$.

A disadvantage of the prime square method is that we cannot recognize in advance whether or not it applies to a specific problem. Theorem 7 suggests the following procedure for solving the system:

(2.26) $$0 < a, b < c,$$

(2.27) $$\alpha < \frac{a}{b} < \beta,$$

(2.28) $$\beta - \alpha < \frac{2}{3c}$$

where $a/b \bmod n$ is computable for all $n$ ($c$ is an integer $> 1$).

ALGORITHM.

*Step* 1. Calculate a prime $p$ with $a/b \equiv 0 \bmod p$.
*Step* 2. Compute $e = a/b \bmod p^2$.
*Step* 3. Choose prime $q$ such that $3c/2p < q < 1/(\beta - \alpha)$.
(This is possible since $1/(\beta - \alpha) > 3c/2$ and $3c/4 \geqq 3c/2p$, and by the well-known Bertrand's postulate there always exists a prime between $3c/4$ and $3c/2$ for $c \geqq 2$.)
*Step* 4. Determine $d = (a/b) \bmod q$.
*Step* 5. Calculate $\alpha'$, $\beta'$, $m$, $m'$ according to the proof of Theorem 7.
*Step* 6. Solve the transformed system (2.22)–(2.25) by the basic algorithm.
*Step* 7. Recalculate $a/b$ from $a'/b'$ as follows: $a/b = p \cdot (b'/a' - 1)$.

*Remark* 1. It should be pointed out that this algorithm is applicable only if the conditions of Theorem 7 are satisfied for some primes $p$ and $q$. See below for the discussion of difficulties.

*Remark* 2. The global condition ($p$ independent) (2.28) should be replaced by (2.17) in Theorem 7 after $p$ has been determined.

*Example*. Assume that Steps 1–4 of the above algorithm have already been performed and we want to solve the following system:

(2.29) $$0 < a, b < 800,$$

(2.30) $$\frac{a}{b} \equiv 0 \bmod 31, \quad \frac{a}{b} \equiv 713 \bmod 961, \quad \frac{a}{b} \equiv 18 \bmod 47,$$

(2.31) $$0.16 < \frac{a}{b} < 0.17.$$

Then Theorem 7 is applicable with $e = 713$, $d = 18$, $p = 31$, $q = 47$; hence $e' = 713/31 = 23$, $(1 + e')^{-1} \equiv 22 \bmod 31$. $d' \equiv 0 \bmod 31$ and $d' \equiv 18 \bmod 47$ yield $d' = 1240$; hence $d'' = 40$ and $(1 + d'')^{-1} \equiv 39 \bmod 47$. The solution of $m' \equiv 22 \bmod 31 \wedge m' \equiv 39 \bmod 47$ yields $m' \equiv 1355 \bmod 1457$ $(=m)$. Further we have $\beta' = 31/31.16 = 0.994865$ and $\alpha' = 31/31.17 = 0.994546$. Thus, we solve the following system:

(2.32) $$0 < a', b' < 1457,$$

(2.33) $$\frac{a'}{b'} \equiv 1355 \bmod 1457,$$

(2.34) $$\frac{1449}{1457} < \frac{a'}{b'} < \frac{1450}{1457}.$$

We obtain $a'/b' = 577/580$; hence $a/b = 93/577$.

*Difficulties*. (1) No prime $p$ is available such that $a/b \equiv 0 \bmod p$. Then apply the transformation $x \to 1 - x$ to the system. See the example below.

(2) There is a $p$ available with $a/b \equiv 0 \bmod p$ but either $\beta - \alpha \geqq 2p/3c$ or $\beta - \alpha < 2p/3c$ and there is no prime between $\beta - \alpha$ and $2p/3c$. Here, we must search for a larger $p$, or the interval length $\beta - \alpha$ must be made smaller.

*Example*. Let us try to solve the following system:

(S1) $$0 < a, b < 800000,$$

(S2) $$\frac{a}{b} \not\equiv 0 \bmod p \quad \text{for all } p < 100000,$$

(S3) $$0.7835 < \frac{a}{b} < 0.7836.$$

We put $a''/b = 1 - a/b$ and find for instance $a''/b \equiv 0 \bmod 499$. We take $p = 499$. Since $c = 800000$ we should have $10^{-4} < 1/q < 998/2400000$, for instance, $q = 3061$. Let us find

(2.35) $$\frac{a''}{b} \equiv 111277 \bmod 499^2,$$

(2.36) $$\frac{a''}{b} \equiv 1729 \bmod 3061.$$

By applying the prime square method we obtain the following system:

(T1) $$0 < a', b' < 1527439,$$

(T2) $$\frac{a'}{b'} \equiv 1290639 \bmod 1527439,$$

(T3) $$\frac{1526776}{1527439} < \frac{a'}{b'} < \frac{1526777}{1527439}.$$

Solving this system we have $a'/b' = 350377/350529$; hence $a''/b = 75848/350377$, and hence $a/b = 274529/350377$.

**2.3. Modulus elimination method.** For solving interval conditioned rational congruences with two or more moduli, we outline another method due to a suggestion of a referee of this paper. Since here the number of moduli is stepwise reduced until a one-modulus system remains, we call it the "modulus elimination method." The system to be solved may be the following one:

$$\frac{a}{b} \equiv e \bmod m, \tag{2.37}$$

$$\frac{a}{b} \equiv f \bmod n, \tag{2.38}$$

$$c < \frac{a}{b} < d, \tag{2.39}$$

$$0 < b < q. \tag{2.40}$$

Assume gcd $(m, n) = 1$, and start by putting

$$t = \left[\frac{c-e}{m}\right], \quad g_1 = t \cdot m + e, \quad g_2 = g_1 + m, \quad h_1 = h_2 = 1.$$

Then we have $g_1 \leqq c$ and $g_2 > c$. Terminate if $g_2 < d$. Otherwise, let

$$j_i = \max\left\{l \mid l \leqq i \text{ and } \frac{g_l}{h_l} \leqq c\right\} \quad \text{and} \quad k_i = \max\left\{l \mid l \leqq i \text{ and } \frac{g_l}{h_l} \geqq d\right\}.$$

Assume that the fractions $g_l/h_l$ have already been interval determined for $1 = l, \cdots, i$, and that none of them satisfy the interval condition (2.39). Then we put

$$g_{i+1} = g_i + g_{j_i}, \quad h_{i+1} = h_i + h_{j_i} \quad \text{if } g_i/h_i \geqq d, \quad \text{or}$$

$$g_{i+1} = g_i + g_{k_i}, \quad h_{i+1} = h_i + h_{k_i} \quad \text{if } g_i/h_i \leqq c.$$

We stop when $g_{i+1}/h_{i+1}$ satisfies the interval condition (2.39). Then we put $g = g_{i+1}$ and $h = h_{i+1}$. Further $u = g_{j_i}$ and $v = h_{j_i}$ when $h_{j_i} < h_{k_i}$, and $u = g_{k_i}$ and $v = h_{k_i}$ otherwise. Finally, we introduce a new fraction $x/y$ by putting

$$a = gx + uy, \qquad b = hx + vy.$$

This leads to $x/y = (av - bu)/(bg - ah)$ and we obtain the reduced congruence system with only modulus:

$$\frac{x}{y} \equiv \frac{vf-u}{g-hf} \bmod n, \tag{2.41}$$

$$c' < \frac{x}{y} < d' \tag{2.42}$$

with transformed interval bounds $c', d'$.

$$0 < hx + vy < q. \tag{2.43}$$

*Example*. The following system has a unique solution:

$$0 < b < 120,$$

$$0.15 < \frac{a}{b} < 0.16,$$

$$\frac{a}{b} \equiv 20 \bmod 23,$$

$$\frac{a}{b} \equiv 22 \bmod 29.$$

According to the above described procedure, we obtain the following sequence of fractions:

$$\frac{-3}{1}, \frac{20}{1}, \frac{17}{2}, \frac{14}{3}, \frac{11}{4}, \frac{8}{5}, \frac{5}{6}, \frac{2}{7}, \frac{-1}{8}, \frac{1}{15}, \frac{3}{22}, \frac{5}{29}, \frac{8}{51}, \ldots$$

where $8/51$ is the first fraction satisfying the interval conditions. The neighbor with the smaller denominator is $3/22$ and so we put

$$a = 3x + 8y \quad \text{and} \quad b = 22x + 51y,$$

which leads to the new system

$$0 < 22x + 51y < 120,$$

$$\frac{-4}{13} < \frac{x}{y} < \frac{7}{6},$$

$$\frac{x}{y} \equiv 1 \bmod 29.$$

Since this system has the unique solution $x = y = 1$ we finally have

$$\frac{a}{b} = \frac{11}{73}.$$

**3. Distribution of *m*-bounded fractions.** The conditions (1.1), (2.1), and (2.2) can be satisfied for at most one irreducible fraction $a/b$ according to Theorem 4, § 2. Therefore, the distance of successive fractions (in natural ordering) of the above type is at least $1/m$, and we are interested in the average distance of all such fractions in the unit interval. To define our problem more precisely we introduce the set $L_m$ of $m$-bounded fractions, and the set $L_{dm}$ of $m$-bounded fractions being congruent $d$ mod $m$. Let

$$G_m = \left\{ \frac{a}{b} \,\middle|\, 0 < a, b < m \wedge \gcd(a, b) = 1 \right\},$$

$$L_m = \left\{ \frac{a}{b} \,\middle|\, \frac{a}{b} \in G_m \wedge \gcd(b, m) = 1 \right\}.$$

The additional condition gcd $(b, m)$ is necessary because $a/b$ mod $m$ is only defined if $b$ has a multiplicative inverse mod $m$. By a theorem of Dirichlet [D49] we have

$$\lim_{m \to \infty} \frac{|G_m|}{m^2} = \frac{6}{\pi^2}. \tag{3.1}$$

By view of the definitions we obviously have $L_m \subseteq G_m$; hence

$$\overline{\lim} \frac{|L_m|}{m^2} \leq \frac{6}{\pi^2}. \tag{3.2}$$

Since for primes $m = p$ we have $L_p = G_p$, we obtain $\lim (|L_p|/p^2) = 6/\pi^2$, which implies equality in (3.2):

$$\overline{\lim} \frac{|L_m|}{m^2} = \frac{6}{\pi^2}. \tag{3.3}$$

The upper limit in (3.3) cannot be replaced by lim because there are sequences of integers $a_1, \cdots$ such that $|L_{a_n}|/a_n^2$ converges to other values than $6/\pi^2$. For instance, we can prove that

$$\lim \frac{|L_{2^n}|}{2^{2n}} = \frac{4}{\pi^2}. \tag{3.4}$$

*Sketch of proof of* (3.4). Let $A_{mb}$ denote the number of prime residues mod $b$ being less than $m$. Then we find

$$|L_m| = m-1+\sum_{\substack{b=2\\ \gcd(b,m)=1}}^{m-1} A_{mb}. \tag{3.5}$$

Since $A_{mb}$ is approximately $m/b\cdot\Phi(b)$ (where $\Phi$ is Euler's function) with $R_{mb} = A_{mb} - m/b\cdot\Phi(b)$ we obtain

$$|L_m| = m-1+\sum_{\substack{b=2\\ \gcd(b,m)=1}}^{m-1} R_{mb}+m\cdot\sum_{\substack{b=2\\ \gcd(b,m)=1}}^{m-1}\frac{\Phi(b)}{b}. \tag{3.6}$$

By view of (3.1), $1/m\cdot\sum_b |R_{mb}|$ converges to zero; hence for $m = 2^n$

$$\lim_{n\to\infty}\left(\frac{|L_{2^n}|}{2^{2n}}-\frac{1}{2^n}\cdot\sum_{\substack{b<2^n\\ b\equiv 1(2)}}\frac{\Phi(b)}{b}\right)=0.$$

It remains to show that

$$\lim_{n\to\infty}\frac{1}{2^n}\cdot\sum_{\substack{b<2^n\\ b\equiv 1(2)}}\frac{\Phi(b)}{b}=\frac{4}{\pi^2}. \tag{3.7}$$

But this follows from (3.1) and the equality

$$B_n-A_n=\frac{1}{2}\cdot B_{n-1}+\frac{1}{4}\cdot A_{n-1} \tag{3.8}$$

where

$$A_n=\frac{1}{2^n}\cdot\sum_{\substack{b<2^n\\ b\equiv 1(2)}}\frac{\Phi(b)}{b}, \tag{3.9}$$

$$B_n=\frac{1}{2^n}\cdot\sum_{b<2^n}\frac{\Phi(b)}{b}. \tag{3.10}$$

As $m$-bounded residue classes $L_{dm}$ we define $L_{dm} = L_m \cap \tilde{Q}_d$, i.e.,

$$L_{dm}=\left\{\frac{a}{b}\,\middle|\,\frac{a}{b}\in L_m \wedge a\equiv bd \bmod m\right\}.$$

Here we will restrict ourselves to prime moduli $m = p$ and we will show that all $L_{dm}$ for $d = 2, \cdots, m-2$ have nearly the same cardinality. More precisely, we prove

$$\lim_{d\to\infty}\lim_{p\to\infty}\frac{|L_{dp}|}{p}=\frac{6}{\pi^2}. \tag{3.11}$$

*Proof of* (3.11). The fractions in $L_{dp}$ have the form

$$\frac{a}{b}=d-\frac{p}{x}\cdot\left[\frac{dx}{p}\right] \quad \text{with } 1\leqq d,x<p. \tag{3.12}$$

For $g = \gcd(x, [dx/p])$ we have $g = 1 \Leftrightarrow \gcd(a, b) = 1$. Therefore we count the integers $x$ with $\lambda p/d \leqq x < (\lambda + 1)p/d$ and $\gcd(x, \lambda) = 1$. By the above $A_{mb}$-notation we then have

$$|L_{dp}| = \sum_{\lambda=1}^{d-1} (A_{(p(\lambda+1)/d),\lambda} - A_{\lambda p/d,\lambda}). \tag{3.13}$$

By view of

$$A_{r,\lambda} = \sum_{t/\lambda} \mu(t) \cdot \left[\frac{r}{t}\right]$$

where $\mu$ denotes Moebius' function, we find from (3.13) that

$$|L_{dp}| = \sum_{\lambda=1}^{d-1} \sum_{t/\lambda} \mu(t) \cdot \left(\left[\frac{p(\lambda+1)}{dt}\right] - \left[\frac{p\lambda}{dt}\right]\right). \tag{3.14}$$

Since for all real numbers $x, y$ we have $x - y - 1 < [x] - [y] < x - y + 1$ we deduce from (3.14) that

$$|L_{dp}| = \sum_{\lambda=1}^{d-1} \sum_{t/\lambda} \mu(t) \cdot \left(\frac{p}{dt} + \eta_{d,p,\lambda,t}\right) \tag{3.15}$$

with $|\eta_{d,p,\lambda,t}| < 1$. This yields

$$|L_{dp}| = \frac{p}{d} \cdot \sum_{\lambda=1}^{d-1} \frac{\Phi(\lambda)}{\lambda} + \frac{\chi_{dp}}{4} \cdot (d^2 + 3d) \tag{3.16}$$

with $|\chi_{dp}| < 1$, since $|\sum_{t/\lambda} \mu(t) \cdot \eta_{d,p,\lambda,t}| \leqq 1 + \lambda/2$. Now, for $p \rightarrow \infty$ we obtain

$$\lim_{p \rightarrow \infty} \frac{|L_{dp}|}{p} = \frac{1}{d} \cdot \sum_{\lambda=1}^{d-1} \frac{\Phi(\lambda)}{\lambda} \tag{3.17}$$

from which our assertion (3.11) follows. □

*Example.* Let $d = 17$, $p = 97$. While $97 \cdot 6/\pi^2 = 58.97$ the exact value for $|L_{17,97}|$ is 57. It turns out that 58 is the most frequent number of $|L_{d,97}|$'s. For 22 $d$-values out of 96 we have $|L_{d,97}| = 58$.

The fractions in $L_{dm}$ are not equally distributed as can be seen in the above example. All 57 fractions $a/b \in L_{17,97}$ lie in the interval $(0, 17]$. Out of these 24 are smaller than one, and for 19 of them we have $[a/b] = 1$; the remaining 14 fractions are greater than two. In fact, when we put

$$K_{dp} = \left\{\frac{a}{b} \middle| \frac{a}{b} \in L_{dp} \wedge a < b\right\}, \tag{3.18}$$

then we can prove

$$|K_{dp}| = \frac{3}{\pi^2} \cdot p + o(p) \qquad (d \neq 1, p-1), \tag{3.19}$$

which says that for greater $p$ half of the fractions of $L_{dp}$ are contained in the interval $[0, 1]$.

*Remark.* Equation (3.19) follows from the statements

$$K_{dp} = \sum_{\lambda=1}^{d-1} (A_{p\lambda/(d-1),\lambda} - A_{p\lambda/d,\lambda})$$

and

$$\lim_{d\to\infty}\lim_{p\to\infty}\frac{K_{dp}}{p}=\lim_{d\to\infty}\frac{1}{d\cdot(d-1)}\cdot\sum_{\lambda=1}^{d-1}\Phi(\lambda)=\frac{3}{\pi^2}\qquad\text{(see [D49]).}$$

**3.1. Expectation values.** Define for each $\lambda = 0, 1, \cdots, p-1$ the set

$$D_{d,p}^{k,\lambda}=\left\{r\,|\,r\in K_{dp}\wedge\frac{\lambda}{p}\leqq r\leqq\frac{\lambda+k}{p}\right\}. \tag{3.20}$$

Then we have

$$E_{d,k}^{p}=\frac{1}{p-k+1}\cdot\sum_{\lambda=0}^{p-k}|D_{d,p}^{k,\lambda}|. \tag{3.21}$$

Since $E_{d,1}^{p}$ is approximately $3/\pi^2$ by (3.19) it can be seen by view of

$$D_{d,p}^{k,\lambda}=\bigcup_{v=\lambda}^{\lambda+k-1}D_{d,p}^{1,v}$$

for $k+\lambda\leqq p$ that

$$\lim_{d\to\infty}\lim_{p\to\infty}E_{dk}^{p}=\frac{3k}{\pi^2}. \tag{3.22}$$

In Table 1 below it is shown that for greater $p$ (3.22) is a very good approximation for the desired expectation values.

Empirically we obtain the following distribution function for $p$-bounded fractions. Let $F(x)$ be the probability that $a/b\leqq x$ under the condition $a/b\in L_{dp}$ with $d$, $p$ fixed. If

$$R_{dpx}=\left\{\frac{a}{b}\,\middle|\,\frac{a}{b}\in L_{dp}\wedge\frac{a}{b}\leqq x\right\},$$

then we have

$$R_{dpx}=L_{d^{-1},p}-R_{d^{-1},p,1/x} \tag{3.23}$$

where $d^{-1}$ means the multiplicative inverse of $d$ mod $p$. By view of (3.23) $F$ satisfies the functional equation

$$F(x)=1-F\left(\frac{1}{x}\right). \tag{3.24}$$

Equation (3.24) yields $F(1) = 0.5$ which is in accordance with (3.19). Further (3.22) suggests the linearity of $F$ for $x\leqq 1$. When we take these facts into account, the following

TABLE 1

| $k$ | $E_{41,k}^{577}$ | $3k/\pi^2$ |
|---|---|---|
| 1 | 0.3033 | 0.3040 |
| 10 | 3.0211 | 3.0396 |
| 100 | 30.0941 | 30.3964 |
| 300 | 90.3489 | 91.1891 |
| 577 | 175.0000 | 175.3870 |

function $F$ reflects very well the actual distribution of the $p$-bounded fractions in $L_{dp}$:

$$(3.25)\qquad \begin{aligned} F(x)&=x/2 && \text{for } 0\leqq x<1,\\ F(x)&=1-1/2x && \text{for } 1\leqq x<d,\\ F(x)&=1 && \text{for } x\geqq d. \end{aligned}$$

The results of this analysis can be summarized as follows.

(1) The generalized residue classes $L_{dp}$ (consisting of the fractions $d$ mod $p$) consist (nearly independent of $d$) of $p\cdot 0.6$ fractions $a/b$ with different denominators less than $p$.

(2) The distribution of the fractions $a/b\in L_{dp}$ is such that half of them are smaller than one, but within the interval [0, 1] they are equally distributed.

(3) The average distance of successive (according to $<$) fractions in $L_{dp}\cap[0,1]$ is $\pi^2/3p$. Therefore, most intervals (more than 80 percent) of length $3/p$ contain exactly one fraction (which can be recovered by Euclidean processes).

**4. Conclusion.** In the case of single modulus systems $x/y\equiv d \bmod m$ we cannot predict how large the modulus $m$ has to be chosen to recover the fraction $x/y$ when we know nothing about the magnitude of numerator $x$ or denominator $y$. Moreover, $x/y$ is not uniquely determined if there are no restrictions for $x$ and $y$. When we have bounds for $x$ and $y$ we can estimate the necessary magnitude of the modulus $m$; but if we want to work with moduli not much greater than $x$ and $y$, then sharp bounds for the fraction $x/y$ itself are required. Here, we suggest essentially two methods for the second of which (PSQM) the range of applicability is not very transparent.

In the case of multiple modulus congruence systems with rational numbers, Matula and Gregory have discovered a solution method using Farey fractions but the seed matrix for their procedure generally contains entries not much smaller than the product of the moduli, and so it does not guarantee an efficient solution to the congruence system. A better approach seems to be the modulus elimination method as well as the intersection method of § 2, where the former has a wider range of applicability but appears to be not quite as efficient as the latter.

**Acknowledgments.** The author thanks the referee who contributed the modulus elimination method for his critical reading of the manuscript. He also expresses thanks to Dr. Johannes Grabmeier for his valuable comments.

## REFERENCES

[D49] L. Dirichlet, *Abhandlungen der Akademie der Wissenschaften*, Berlin, 1849.

[GK84] R. T. Gregory and E. V. Krishnamurthy, *Methods and Applications of Error-Free Computation*, Springer-Verlag, Berlin, New York, 1984.

[Kn81] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, Addison-Wesley, Reading, MA, 1981.

[Pe58] O. Perron, *Theorie der Kettenbrueche*, Teubner-Verlag, Stuttgart, 1958.

[Sc66] A. Schoenhage, *Multiplikation grosser Zahlen*, Computing, 1 (1966), pp. 182–196.

[SS71] A. Schoenhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing, 7 (1971), pp. 281–292.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 91-99, February 1989

# HOW LONG CAN A EUCLIDEAN TRAVELING SALESMAN TOUR BE?*

HOWARD J. KARLOFF†

**Abstract.** Where $S$ is a set of $N$ points in the unit square, let $t(S)$ be the Euclidean length of the shortest traveling salesman tour through $S$. Let $t_N = \max_S t(S)$. We improve Few's bound [*Mathematika*, 2 (1955), pp. 141-144] that $t_N \leqq \sqrt{2N} + 7/4$ to $t_N \leqq \alpha\sqrt{N} + 11$, where $\alpha/\sqrt{2} < 0.984$.

**Key words.** Euclidean traveling salesman, Strips algorithm, Euclidean minimum spanning tree, Euclidean perfect matching

**AMS(MOS) subject classifications.** 05C45, 51M05, 68R10

**1. Introduction.** If $n$ points are placed into the unit square

$$\{(x,y) \mid 0 \leqq x \leqq 1, 0 \leqq y \leqq 1\},$$

how long can the shortest Euclidean traveling salesman tour through those $N$ points be? (A *Euclidean traveling salesman tour* is a closed walk passing through all the points; its length is measured by the Euclidean metric.) Let $t_N$ be the maximum possible value (by continuity and compactness, $t_N$ exists). In 1955, Few [4] proved the best upper bound known on $t_N$: $t_N \leqq \sqrt{2N} + 7/4$; the best lower bound known, $t_N \geqq (4/3)^{1/4}\sqrt{N} - o(\sqrt{N})$ ($(4/3)^{1/4} < 1.075$) is obtained by approximately tessellating the unit square with equilateral triangles. This gap—$(4/3)^{1/4}$ is about 24 percent smaller than $\sqrt{2}$—has stood since 1955. (A $\sqrt{2N} + o(\sqrt{N})$ bound was rediscovered in 1983 by Supowit, Reingold, and Plaisted [7].) We break the $\sqrt{2}$ barrier by proving that $t_N \leqq \alpha\sqrt{N} + 11$, where $\alpha/\sqrt{2} < 0.984$.

Let us define a tour to be *Manhattan* if it uses only segments parallel to the square's sides (but there may be "corners" anywhere, not only at the given $N$ points). It is possible to place $N - o(N)$ points in the unit square so that the Manhattan (i.e., $L^1$) distance between every pair is at least $\sqrt{2}/\sqrt{N}$—a grid rotated 45° will do—and therefore every Manhattan tour must have length at least $\sqrt{2}\sqrt{N} - o(\sqrt{N})$; consequently any method that decreases the constant must, as does ours, generate some non-Manhattan tours. (Few's method generates Manhattan tours.) Methods that generate non-Manhattan tours are harder to analyze.

Researchers have also studied related problems. Chung and Graham [3] improved Few's upper bound on the length of the shortest Steiner tree through $N$ points in the unit square from $\sqrt{N} + 7/4$ to $0.995\sqrt{N}$. (They gave details for an improvement to $0.99995\sqrt{N}$ only.) Moran [6] studied the maximum value of the shortest traveling salesman tour through a set of $N$ points of unit diameter in $\mathbb{R}^k$. Our technique is related to these authors' methods. In § 5 we will show how our result improves the bounds for the minimum spanning tree and minimum weight perfect matching problems.

(A different, more popular line of research deals with the *average* values of these quantities, when, say, $N$ points are uniformly drawn from the unit square, and either the Euclidean or $L^1$ metric is used. See, e.g., [5], [2], [1].)

**2. Preliminaries and preprocessing.** Our bound will be proven by exhibiting an algorithm that generates the desired tour; the algorithm will use infinite-precision real numbers. Modeled after Few's original Strips method, the algorithm uses two key ideas.

* Received by the editors February 9, 1988; accepted for publication (in revised form) June 13, 1988. This work was supported in part by the National Science Foundation under grant DCR 8609733.

† Computer Science Department, University of Chicago, 1100 East 58th Street, Chicago, Illinois 60637.

(1) We can preprocess the points by iteratively replacing a pair of points that are very near each other by their midpoint, and patch in the pair later;

(2) Then, if no two points are too close, a local modification of Strips will achieve the bound.

First, we give some notation. If $X$ and $Y$ are points in the plane, $XY$ will denote both the line segment between $X$ and $Y$ and its Euclidean length. If $X_1, X_2, \cdots, X_k$ are points, $X_1X_2\cdots X_k$ will denote both the walk from $X_1$ to $X_k$ formed from segments $X_1X_2$, $X_2X_3, \cdots, X_{k-1}X_k$, and its length, $X_1X_2 + X_2X_3 + X_3X_4 + \cdots + X_{k-1}X_k$.

We need a lemma.

LEMMA 1. *Let $S$ be a set of $n \geqq 3$ points in the Euclidean plane. Let $D = \min AB$, where the minimum is over all pairs of distinct points in $S$, and let $XY = D$, where $X$, $Y \in S$. Let $M$ be the midpoint of $XY$. Choose $Z$, $W \in S - \{X, Y\}$. Then*

$$\min\{ZX+XY+YW, ZY+YX+XW\}-(ZM+MW)\leqq(3-\sqrt{3})D.$$

Lemma 1 appeared in a slightly different form from Lemma 4.1 in [6]. For completeness, the reader can find a proof of Lemma 1 in the Appendix.

In the preprocessing phase, we repeatedly replace the pair of nodes closest together by their midpoint, until no pair are too close together. The value of the parameter $c$ ($\sqrt{10}/6 \leqq c \leqq \sqrt{10}/4$) that appears in line (2) of the preprocessing algorithm will be determined later.

Here is the preprocessing phase, for a set $S_N$ of $N$ points:

```
(1) $k := N$.
(2) Let $D_k = c/\sqrt{k}$.
(3) Let $D$ be the minimum distance between two points in $S_k$.
    If $D > D_k$ or if $|S_k| < 4$, assign $n := k$ and halt.
(4) Let $X_k$, $Y_k \in S_k$ such that $X_kY_k = D$.
    Let $M_k$ be the midpoint of $X_kY_k$.
    $S_{k-1} = (S_k - \{X_k, Y_k\}) \cup \{M_k\}$ /* $|S_{k-1}| \leqq k-1$. */
    $k := k-1$.
(5) Go to 2.
End
```

After applying the preprocessing phase, we use a modified version of Strips (to be described later) to find a tour $T_n$ through the at most $n$ points in $S_n$. Then we patch in the deleted points:

```
For $k := n+1$ to $N$ do
    In tour $T_{k-1}$, let $Z$ and $W$ be $M_k$'s predecessor and successor.
    To get $T_k$, replace the portion $Z-M_k-W$ in $T_{k-1}$ by either $Z-X_k-Y_k-W$
        or $Z-Y_k-X_k-W$, whichever is shorter.
```

THEOREM 1. $[($*length of* $T_N) - ($*length of* $T_n)] \leqq 2c(3-\sqrt{3})(\sqrt{N}-\sqrt{n})$.

*Proof.* Let $n+1 \leqq k \leqq N$ and let $D = X_kY_k$. By the choice of $X_k$, $Y_k$, $D = \min AB \leqq D_k$, where the minimum is over all pairs of distinct nodes in $S_k$. $|S_k| \geqq 3$. By Lemma 1,

$$\min\{ZX_k+X_kY_k+Y_kW, ZY_k+Y_kX_k+X_kW\}-(ZM_k+M_kW)$$
$$\leqq(3-\sqrt{3})D\leqq(3-\sqrt{3})D_k.$$

Thus [(length of $T_k$) − (length of $T_{k-1}$)] $\leqq (3-\sqrt{3})D_k$. Thus

$$\begin{aligned}[(\text{length of } T_N)-(\text{length of } T_n)] &= \sum_{k=n+1}^{N} [(\text{length of } T_k)-(\text{length of } T_{k-1})] \\ &\leqq \sum_{k=n+1}^{N} (3-\sqrt{3})D_k = \sum_{k=n+1}^{N} (3-\sqrt{3})\frac{c}{\sqrt{k}} \\ &\leqq c(3-\sqrt{3})\int_{x=n}^{x=N} x^{-1/2}dx = c(3-\sqrt{3})(2x^{1/2})\,|_{x=n}^{x=N} \\ &= 2c(3-\sqrt{3})(\sqrt{N}-\sqrt{n}). \qquad \square\end{aligned}$$

Theorem 4.2 in [6] and our Theorem 1 are very similar.

**3. The main procedure.** Let $S = S_n$ be a set of at most $n$ points in the unit square, no two separated by distance $D_n = c/\sqrt{n}$, or less. Because $|S| = n$ in the worst case, we will assume this without loss of generality. First we present the Strips algorithm, which is the starting point for our algorithm (actually, we have modified it and its analysis slightly).

Our version of Strips generates two tours, the shorter of which is guaranteed to have length $\leqq \sqrt{2n} + 11$. Here is a naive version generating only one tour. Let $w = \sqrt{2}/\sqrt{n}$ and $w' = D_n/\sqrt{5} = (c/\sqrt{5})(1/\sqrt{n}) = (c/\sqrt{10})w$. Divide the unit square into $\lceil 1/w \rceil$ horizontal nonoverlapping contiguous $w \times 1$ strips, each of width $w$, enlarging the square vertically by less than $w$ if necessary. Now extend the square to the right by at most $w'$, if necessary, to ensure that the horizontal width is a multiple of $w'$. Mark each width-$w$ strip's horizontal center line with a dotted line. Follow the uppermost strip's center line rightward, starting at its left end, jogging vertically (and only vertically) to reach each point in the strip, and then immediately returning to the center line; the incremental cost due to a point in the strip is double its vertical distance from the center line. At the right end, drop down to the right end of the second strip's center line, and follow it leftward, jogging up and down to visit each point in the strip. Changing direction with each strip, repeat this process until the bottom strip is crossed. Returning to the starting point adds at most $2 + w'$ to the total length, even if only horizontal and vertical edges are used.

Note that a Strips tour is a Manhattan tour.

Now we present the improved version of Strips. Run Strips a second time, this time using the original center lines as boundaries for a set of $1 + \lceil 1/w \rceil$ new strips of width $w$, each displaced vertically $w/2$ from a previous strip (the top and bottom ones can be viewed as width-$w$ strips whose upper and lower halves, respectively, contain no points); the horizontal center lines of these new strips are the old strip boundaries. Together, over *both* Strips runs, each point contributes a vertical cost of $w$. Thus the total cost, over both runs, is

$$\leqq nw + 2\left[\left(1+\left\lceil\frac{1}{w}\right\rceil\right)(1+w')+(1+w')+2(1+w)\right] \leqq nw + \frac{2}{w} + 22.$$

$w = \sqrt{2/n}$ implies that the total cost is at most $\sqrt{2n} + \sqrt{2n} + 22$. Thus the length of the shorter tour is at most $\sqrt{2n} + 11$.

Our procedure Local_Strips divides each horizontal strip into $\lceil 1/w' \rceil$ disjoint $w \times w'$ rectangular regions (as does Strips), extending the right boundary of the unit square by at most $w'$ if necessary; points on a boundary are arbitrarily assigned to one region.

FACT. *If* $c \geqq \sqrt{10}/6$ *and we divide a* $w \times w'$ *rectangle into a vertically centered* $2w' \times w'$ *rectangle, and upper and lower* $((w/2) - w') \times w'$ *rectangles, then the central*

*$2w' \times w'$ rectangle is the largest, and none of the three rectangles can contain more than one point.*

*Proof.* The diagonal of a $2w' \times w'$ rectangle is of length $w'\sqrt{5} = D_n$. The rest is trivial. □

Here is a description of Local_Strips. Local_Strips constructs two tours; the first is based on the original $\lceil 1/w \rceil$ strips, and the second on the $1 + \lceil 1/w \rceil$ displaced strips. The smaller is suitably short. As in Strips, in each strip Local_Strips starts at one end of the center line and moves horizontally toward the other. The difference is what happens *within* each strip's $\lceil 1/w' \rceil$ $w \times w'$ regions. Local_Strips starts at one side's midpoint $G$ and ends at the other side's midpoint $H$. *Within the region, it visits the three or fewer nodes in an optimal way.* In other words, among the six or fewer $G - H$ paths passing through all the points in the $w \times w'$ region, Local_Strips uses the shortest. This is the only difference between Few's Strips and Local_Strips: in each of the $w \times w'$ rectangular regions in a strip, Local_Strips visits all the nodes optimally (starting at the midpoint of one side and ending at the other side's midpoint), while Strips blindly moves horizontally, jogging vertically to reach each point.

Note that a Local_Strips-generated tour is probably not Manhattan. The idea of using non-Manhattan modifications to a Manhattan tour appeared as early as 1950 [8], and thus predates Few's paper.

**4. Analysis of Local_Strips.** We will analyze the sum of the lengths of the two tours generated by Local_Strips. Rather than analyzing it directly, we will calculate the savings over the version of Strips described earlier.

First, we need a few lemmas. In $\mathbb{R}^2$, let $I$ be the closed interval between $(0, 1)$ and $(1, 1)$, let $J$ be the one between $(0, 0)$ and $(1, 0)$, and let $K$ be the one between $(0, -1)$ and $(1, -1)$.

LEMMA 2. *Let $P = (0, 0)$, $Q = (1, 0)$. For a finite $W \subset \mathbb{R}^2$, let $g(W)$ be the length of the shortest walk that begins at $P$, ends at $Q$, and passes through all the points in $W$. Then*

$$\max_{A \in I,\, B \in J,\, C \in K} g(\{A, B, C\}) = \frac{31 + \sqrt{2}}{7} < 4.631.$$

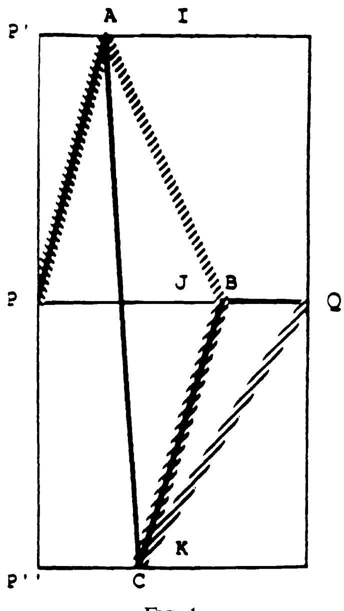


FIG. 1

*Proof*. Without loss of generality, $A$ is the leftmost point.

*Case* 1. The left-to-right order is $A$, $C$, $B$. See Fig. 1.

SUBLEMMA. *Among the six paths $PABCQ$, $PACBQ$, $PBACQ$, $PBCAQ$, $PCABQ$, $PCBAQ$, either the first or second is the shortest.*

*Proof of Sublemma*. Both $PBACQ$ and $PBCAQ$ self-intersect and it is known that the shortest $P-Q$ path does not self-intersect.

$$\begin{aligned}PC \geqq PA, AQ \geqq CQ \Rightarrow (PC+AQ)+CB+AB &\geqq (PA+CQ)+CB+AB\\ \Rightarrow PCBAQ &\geqq PABCQ.\end{aligned}$$

Now $PC \geqq PA$, $AB \geqq CB$ implies that

$$PC+CA+AB+BQ \geqq PA+AC+CB+BQ,$$

so that $PCABQ \geqq PACBQ$. □

For now, let us assume that $A=(0, 1)$, $C=(0, -1)$, $B=(b, 0)$ $(0 \leqq b \leqq 1)$ and let us find $b=b^*$ to maximize $g(\{A, B, C\})$:

$$g(\{A,B,C\}) = \min(1+2+\sqrt{b^2+1}+1-b, 1+2\sqrt{b^2+1}+\sqrt{2}).$$

It is not hard to see that the maximum occurs at $b^*=(9-4\sqrt{2})/7$, where

$$\begin{aligned}4-b^*+\sqrt{1+(b^*)^2} &= 1+\sqrt{2}+2\sqrt{1+(b^*)^2}\\ &= \frac{31+\sqrt{2}}{7} \in (4.630, 4.631).\end{aligned}$$

Having finished that, let us allow $A$, $B$, $C$ to vary again (but stay in Case 1). Fix $A$ on $I$, $B$ on $J$. Project $A$ and $B$ down to $K$, getting $A'$ and $B'$, respectively; allow $C$ to vary between them. Let

$$\begin{aligned}r &= AA'+A'B = 2+A'B,\\ s &= AB'+B'B = AB'+1;\\ AB' &\leqq 1+A'B \Rightarrow r \geqq s.\end{aligned}$$

But $\{X \in \mathbb{R}^2 | AX+XB \leqq r\}$ is an ellipse that contains $A'$ and $B'$ and by convexity of ellipses, the segment between them. Thus, for $C$ between $A'$ and $B'$, $AC+CB \leqq AA'+A'B$. Also $BC+CQ \leqq BA'+A'Q$, for $C$ in the same range. It follows that there is an optimal Case 1 configuration with $C=A'$.

Where $B=(b, 0)$, now allow $A$ to vary on $I$ between $P'=(0, 1)$ and $(b, 1)$. Let $C=A'$ be the projection of $A$, as before. Let $P''=(0, -1)$:

$$\begin{aligned}PACBQ &= PA+AA'+A'B+BQ\\ &= PA+AB+(2+BQ)\\ &\leqq PP'+P'B+(2+BQ) = PP'P''BQ;\end{aligned}$$

by a similar convexity argument,

$$PABCQ = PA+AB+BA+AQ.$$

But $PA+AB \leqq PP'+P'B$ and $BA+AQ \leqq BP''+P''Q$. Thus $PABCQ \leqq PP'BP''Q$. Therefore, there is a Case 1 optimum with $A=P'$, $C=P''$. We know that in this case the maximum is $(31+\sqrt{2})/7$ when $B=((9-4\sqrt{2})/7, 0)$.

The rest is easy.

*Case* 2. The left-to-right order is $A$, $B$, $C$.

Let $Q''=(1, -1)$. Consider only the path $PABCQ$. Fix $B$ and allow $A$ and $C$ to vary (but stay in Case 2). By a convexity argument, we may assume $A=P'$ and $C=Q''$.

Now, allowing $B$ to vary, we see that we may assume $B = P$. Thus the length of $PABCQ$ in Case 2 is at most $PP' + P'P + PQ'' + Q''Q = 3 + \sqrt{2} < 4.63$. □

We state Lemma 3 without proof. This lemma will enable us to bound the improvement gained over Few's algorithm by ours.

LEMMA 3. (1) *If $A \in I$, $B \in J$, $C \in K$, then* $g(\{A, B, C\}) \leqq (31 + \sqrt{2})/7 = 5 - ((4 - \sqrt{2})/7)$.

(2) *If $A \in I$, $C \in K$,* $g(\{A, C\}) \leqq 3 + \sqrt{2} < 5 - 0.5$.

(3) *If $A \in I$, $B \in J$,* $g(\{A, B\}) < 1 + (\sqrt{2}/2) + \sqrt{(5/2)\sqrt{2}} < 3 - 0.25$.

(4) *If $A \in I$,* $g(\{A\}) \leqq 1 + \sqrt{2} < 3 - 0.25$.

*In cases* (1) *and* (2), *there are two points on $I \cup K$ and* $((1 + 2\cdot 2) - (g\ value))/2 \geqq (4 - \sqrt{2})/14$. *In cases* (3) *and* (4), *there is one point on $I \cup K$ and* $((1 + 2\cdot 1) - (g\ value))/1 > (4 - \sqrt{2})/14$.

Now the analysis of Local_Strips is simple. Within each $w \times w'$ rectangle, consider the central $2w' \times w'$ subrectangle, and the upper and lower $((w/2) - w') \times w'$ subrectangles. Each contains at most one point.

DEFINITION. If $X$ is a point in an $w \times w'$ rectangle $R$, let $d(X)$ be the signed vertical distance between $X$ and $R$'s horizontal center line (positive if $X$ is above the center line, negative if below).

Let $T$ be the set of at most three points in $R$. Then the Strips cost associated with the points $T$ in $R$ is

$$w' + 2 \sum_{X \in T} |d(X)|.$$

Where $G$ and $H$ are the midpoints of the vertical sides of the $w \times w'$ rectangle, let $h(T)$ be the length of the shortest $G - H$ path containing all the nodes in $T$—this is the Local_Strips cost for $R$. Our goal is to establish a positive lower bound on

$$\left[w' + 2 \sum_{X \in T} |d(X)|\right] - h(T),$$

the savings we obtain over Strips.

THEOREM 2. *Let $T$ be a set of points in a $w \times w'$ rectangle such that at most one has $d(X) \geqq w'$, at most one has $d(X) \leqq -w'$, and at most one has $|d(X)| < w'$. Let $k$ be the number of $X \in T$ such that $|d(X)| \geqq w'$. ($k \in \{0, 1, 2\}$.) Then if $k \in \{1, 2\}$,*

$$\frac{[w' + 2 \sum_{X \in T} |d(X)|] - h(T)}{k} \geqq \frac{4 - \sqrt{2}}{14} w'.$$

*Proof*. By the triangle inequality, $[w' + 2 \sum_{X \in T} |d(X)|] - h(T)$ cannot increase if $|d(X)|$ decreases for any $X \in T$. Thus we may assume that

$$d(X) \geqq w' \Rightarrow d(X) = w',$$
$$d(X) \leqq -w' \Rightarrow d(X) = -w', \quad \text{and}$$
$$|d(X)| < w' \Rightarrow d(X) = 0.$$

In other words, the points lie somewhere on the top, center, or bottom segments of a $2w' \times w'$ rectangle. Without loss of generality, we may assume that there is a point $X \in T$ with $d(X) = w'$.

Now we use Lemma 3, which applied to $2 \times 1$ rectangles. If $k = 2$, we are in case (1) or (2) and then $w' + 2 \sum_{X \in T} |d(X)| = 5w'$. Cases (1) and (2) of Lemma 3 imply that

$$\frac{[w'+2\sum_{X\in T}|d(X)|]-h(T)}{k} \geqq w'\left(5-\frac{31+\sqrt{2}}{7}\right)\frac{1}{2} = w'\left(\frac{4-\sqrt{2}}{14}\right).$$

If $k = 1$, $w' + 2\sum_{X\in T}|d(X)| = 3w'$ and cases (3) and (4) imply that

$$\frac{[w'+2\sum_{X\in T}|d(X)|]-h(T)}{k} > \frac{w'}{4} > w'\left(\frac{4-\sqrt{2}}{14}\right). \qquad \square$$

THEOREM 3. *Choose $c$ and $\alpha$ to satisfy*

$$2(3-\sqrt{3})c=\alpha, \qquad \alpha=\sqrt{2}-\left(\frac{4-\sqrt{2}}{28\sqrt{5}}\right)c.$$

*Thus $c \in (0.548, 0.549)$ (so that $\frac{1}{6} < c/\sqrt{10} < \frac{1}{4}$) and $\alpha < 1.3916$ (and $\alpha/\sqrt{2} < 0.984$). Then the length of the shorter Local_Strips tour is at most $\alpha\sqrt{n} + 11$.*

*Proof.* Let $K$ be the number of points at a distance at least $w'$ away from the center lines of the original strips, and let $K'$ be the number at a distance at least $w'$ away from the center lines of the displaced strips. We make the following crucial observation. Since $c \leqq \sqrt{10}/4 \Rightarrow w' + w' \leqq w/2$, every point contributes to either $K$ or $K'$ (of course some may contribute to both). Thus $K + K' \geqq n$.

By Theorem 2, the savings achieved by the first Local_Strips tour over the first Strips tour is at least $K\cdot w'(4-\sqrt{2})/14$; the savings achieved by the second Local_Strips tour over the second Strips tour is at least $K'\cdot w'(4-\sqrt{2})/14$. Since $K + K' \geqq n$, the total length of the two Local_Strips tours is at most

$$\begin{aligned}(2\sqrt{2n}+22)-\frac{n}{14}w'(4-\sqrt{2}) &= 22+\sqrt{n}\left(2\sqrt{2}-\frac{(4-\sqrt{2})}{14}\frac{c}{\sqrt{5}}\right)\\ &= 22+2\sqrt{n}\left(\sqrt{2}-\frac{(4-\sqrt{2})}{28\sqrt{5}}c\right)\\ &= 22+2\alpha\sqrt{n},\end{aligned}$$

and therefore the shorter one has length at most $\alpha\sqrt{n} + 11$. $\square$

THEOREM 4. *The length of the shorter tour through $S_N$ is at most $\alpha\sqrt{N} + 11$.*

*Proof.* The proof is immediate from Theorems 1 and 3 and the fact that $2c(3-\sqrt{3}) = \alpha$. $\square$

**5. Conclusions.** We have improved Few's bound by more than 1.6 percent. Obtaining a slightly better improvement—at the cost of complicating the algorithm—is not too difficult. For example, many points may contribute to both the $K$ and $K'$ of the proof of Theorem 3, yet we exploit only the fact that $K + K' \geqq n$. A second potential method is to prove a stronger version of Theorem 2—as it stands, in the worst case configuration of Lemma 3(1), the three points $A$, $B$, and $C$ all lie within distance $D_n$ of each other. Probably the most promising method is to drop Lemmas 2 and 3 altogether, and not divide the horizontal strips into vertical regions at all, but instead simply to run $L^2$-Strips in each strip; $L^2$-Strips is the variant of Few's algorithm that visits the points in the same order but runs *directly* from one point to the next, using diagonal edges. We should be able to bound the total length favorably as long as no points are within a distance $D_n$ or less from each other.

Small improvements in the constant are probably uninteresting. However, improving the bound by, say, 10 percent, would require interesting new ideas.

This traveling salesman improvement immediately implies improvements in the known bounds for minimum spanning tree (MST) and minimum weight perfect matching (for even $N$) in the unit square. A tour minus one edge is a spanning tree; thus the MST on $N$ points has length at most $\alpha\sqrt{N} + 11$. If $N$ is even, then alternate edges of a tour form a perfect matching. Choosing the better of the two implied perfect matchings, we infer immediately that the length of a minimum weight perfect matching is at most $(\alpha/2)\sqrt{N} + (11/2)$.

To prove that the lower bound based on the tessellation by equilateral triangles is tight (for any of these three problems), we might consider, not a unit square, but an equilateral triangle with unit-length sides. Let the $l$th *triangular number* be $l(l+1)/2$. If $N$ is the $(2^k + 1)$st triangular number $(2^k+1)(2^k+2)/2$, the equilateral triangle can be tessellated with $N$ points perfectly—there are no boundary effects, as there are in the cube—and this might aid in a proof that the tessellation is precisely the worst case. Probably the MST problem is the easiest here. With MST, we might be able to use a simple MST algorithm to deduce the configuration that results in the longest MST.

**Appendix. Proof of Lemma 1.** Let $U$ be the union of the open disks of radius $D$ about $X$ and $Y$. (See Fig. 2.) No points of $S$ other than $X$ and $Y$ lie in $U$. Note that

$$\begin{aligned}\min\{ZX+XY+YW, ZY+YX+XW\}-(ZM+MW)\\ \leq \frac{ZX+XY+YW+ZY+XY+XW}{2}-(ZM+MW)\\ =D+\frac{g(Z)+g(W)}{2},\end{aligned}$$

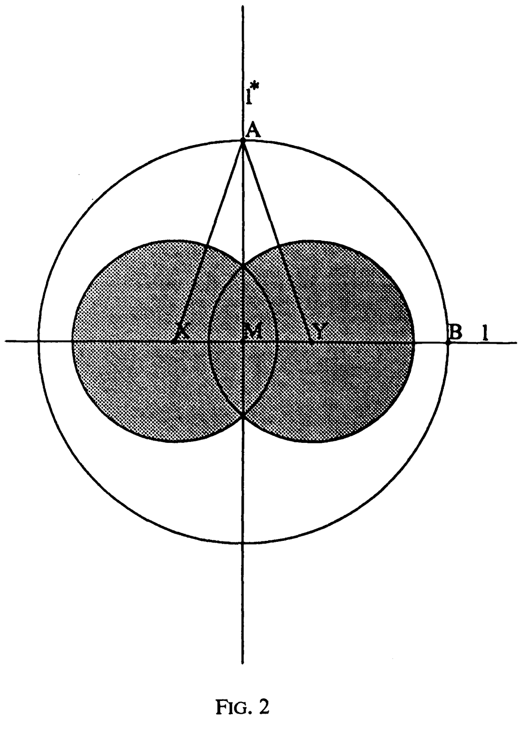


FIG. 2

where $g(P) = PX + PY - 2PM$. We will show that if $P$ is not in $U$, $g(P) \leqq 2D - D\sqrt{3}$, thereby proving the lemma.

Fix $r \geqq 0$ and let us try to maximize $g(P)$ over all $P$ such that $PM = r$. It is not difficult to show that every point at distance less than $(\sqrt{3}/2)D$ from $M$ lies in $U$; thus we may assume $r \geqq (\sqrt{3}/2)D$. Let $l$ be the (infinite) line through $X$ and $Y$ and let $l^*$ be the line perpendicular to $l$ that passes through $M$. Let $A$ be a point on $l^*$ such that $AM = r$ and let $B$ be a point on $l$ such that $BM = r$. Then $BX + BY = 2BM = 2r$. Let $d = AX + AY > 2r$.

Since $BX + BY < d$ and $AX + AY = d$, the ellipse $\{C \in \mathbb{R}^2 \mid CX + CY \leqq d\}$ contains $A$ and $B$ and therefore every point on the disk $\{E \in \mathbb{R}^2 \mid EM \leqq r\}$. Thus $PM = r \Rightarrow PX + PY \leqq d$. Furthermore, the intersection of $\{C \mid CX + CY = d\}$ and $\{C \mid CM = r\}$ consists of two points alone, $A$ and $A$'s reflection through $l$. Thus given $r$, the worst case occurs on $l^*$.

Yet decreasing $PM$ cannot decrease $PX + PY - 2PM$. Consequently the maximum occurs when $r = (\sqrt{3}/2)D$ and $P$ lies on the common intersection of $l^*$ and the two circles of radius $D$ about $X$ and $Y$. There,

$$g(P) = D + D - 2\left(\frac{\sqrt{3}}{2}D\right) = 2D - D\sqrt{3}.$$

Thus $g(P) \leqq 2D - D\sqrt{3}$ for all $P$ not in $U$. □

**Acknowledgments.** Joan Boyar and Barry Merriman helped to prove the crucial Lemma 2, which the author originally thought would be too hard to prove. Without the numerous simplifications suggested by Peter Hajnal, this paper would have been far more complicated. To Peter, who was instrumental in proving Lemma 1, the author owes a debt of gratitude. The author also thanks Jeff Shallit and Bob Lord for their help in drawing the figures and the anonymous referees for their helpful suggestions.

## REFERENCES

[1] M. Bern, *Two probabilistic results on rectilinear Steiner trees*, in Proc. 18th Annual ACM Symposium on the Theory of Computing, Berkeley, CA, 1986, pp. 433–441.

[2] J. Beardwood, J. H. Halton, and J. M. Hammersley, *The shortest path through many points*, Proc. Cambridge Philos. Soc., 55 (1959), pp. 299–327.

[3] F. R. K. Chung and R. L. Graham, *On Steiner trees for bounded point sets*, Geom. Dedicata, 11 (1981), pp. 353–361.

[4] L. Few, *The shortest path and shortest road through n points*, Mathematika, 2 (1955), pp. 141–144.

[5] R. M. Karp and J. M. Steele, *Probabilistic analysis of heuristics*, in The Traveling Salesman Problem, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., John Wiley, New York, 1985, pp. 181–205.

[6] S. Moran, *On the length of optimal* TSP *circuits in sets of bounded diameter*, J. Combin. Theory Ser. B, 37 (1984), pp. 113–141.

[7] K. J. Supowit, E. M. Reingold, and D. A. Plaisted, *The traveling salesman problem and minimum matching in the unit square*, SIAM J. Comput., 12 (1983), pp. 144–156.

[8] S. Verblunsky, *On the shortest path through a number of points*, Proc. Amer. Math. Soc., 2, 1951, pp. 904–913.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 100-107, February 1989

# A LINEAR REORDERING ALGORITHM FOR PARALLEL PIVOTING OF CHORDAL GRAPHS*

JOSEPH W. H. LIU† AND ANDRANIK MIRZAIAN†

**Abstract.** This paper provides an efficient algorithm for generating an ordering suitable for the parallel elimination of nodes in chordal graphs. The time complexity of the reordering algorithm is shown to be linear in the size of the chordal graph. The basic parallel pivoting strategy is originally by Jess and Kees [*IEEE Trans. Comput.*, C-31 (1982), pp. 231-239]. The relevance of the reordering to parallel factorization of sparse matrices (not necessarily chordal) is also discussed.

**Key words.** chordal graph, sparse matrix, parallel pivoting, zero deficiency, sparse elimination

**AMS(MOS) subject classifications.** 65F50, 65F25

## 1. Introduction.

In this paper, we consider the *parallel pivoting* problem, which arises in the exploitation of parallelism in the direct solution of large sparse symmetric positive definite linear systems. For a given large sparse symmetric matrix $A$, we want to determine an ordering which is appropriate in terms of preserving sparsity and exploiting parallelism in its Cholesky factorization. Parallel pivoting strategies have been studied by Alaghband and Jordan [1], Betancourt [2], Calahan [3], Huang and Wing [7], Jess and Kees [8], Peters [10], and others.

A modular approach has been used by Jess and Kees [8], which determines a good fill-reducing ordering $P$ for $A$, and then finds an equivalent reordering $\tilde{P}$ (that is, one that preserves the filled graph) suitable for parallel elimination. Since $PAP^t$ and $\tilde{P}A\tilde{P}^t$ have the same set of fills, $\tilde{P}$ has the same fill-reducing property as $P$. Moreover, it is well known that the filled graph of $PAP^T$ (that is, the graph of $G(PAP^T)$ together with the fill edges due to factorization) is *chordal* [11]. Therefore, the problem is reduced to finding a perfect elimination ordering for a chordal graph, which is appropriate for parallel elimination. In [8], Jess and Kees have also provided such a parallel pivoting strategy for chordal graphs. It is shown in [9] that the resulting reordering has the desirable property of minimizing the number of parallel elimination steps among the class of perfect orderings (orderings with no file).

In this paper, we provide an efficient algorithm to generate this parallel pivoting sequence. Central to the algorithm is an effective method to identify nodes with *zero deficiency* in a given chordal graph. The method is based on an interesting property of perfect elimination orderings. Repeated use of this zero-deficiency test results in an overall reordering algorithm. We prove that the time complexity of this new algorithm is linear with respect to the number of nodes and number of edges in the chordal graph.

The reader is assumed to be familiar with basic notions related to chordal graphs. The book by Golumbic [6] contains an excellent treatment of the subject. Moreover, graph-theoretic notions relevant to sparse matrix computation are also assumed. The reader is referred to George and Liu [5].

An outline of this paper follows. In § 2, the parallel pivoting strategy by Jess and Kees [8] for chordal graphs is reviewed. In § 3, we provide a simple, efficient zero-

* Received by the editors April 6, 1987; accepted for publication July 28, 1988. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under grants A5509 and A5516, the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc., and by the U.S. Air Force Office of Scientific Research under contract AFOSR-ISSA-86-00012.

† Department of Computer Science, York University, North York, Ontario, Canada M3J 1P3.

deficiency test on nodes in a given chordal graph. We apply this result in § 4 to obtain a linear time reordering algorithm that will produce a desirable parallel elimination sequence. Section 5 contains the concluding remarks.

## 2. Review of parallel elimination algorithm by Jess and Kees.

**2.1. The parallel elimination algorithm.** A graph is called *chordal* (or *triangulated*) if every cycle of length 4 or more has a chord; or equivalently, if it has a *perfect elimination ordering* [6]. It is known that a perfect elimination ordering of a chordal graph can be found in linear time [6], [12]. In [8], Jess and Kees provide a reordering scheme tailored for parallel elimination of a chordal graph. In [9], a minor variant of their strategy is shown to produce orderings with minimum number of parallel node elimination steps among all perfect elimination orderings. It is indeed a relevant scheme for parallel elimination.

In this section, we briefly review the Jess and Kees' strategy in preparation for our algorithm in the next section. We first introduce some terminology. Let $A$ be a given $n$-by-$n$ sparse symmetric positive definite perfect elimination matrix, that is, its graph $G(A)$ is chordal. We shall use $G(A)$ and $G$ interchangeably to refer to this chordal graph. For convenience, we shall also use $G$ to refer to the set of nodes in this graph.

Two nodes are said to be *independent* if they are not adjacent. A node is said to have *no* (or *zero*) *deficiency* if its adjacent set is a clique [11]. In the literature, such a node is also referred to as *simplicial* [6]. It is well known that a chordal graph always has one node with no deficiency. Furthermore, if the graph is not a clique, there are at least two such independent nodes. Jess and Kees make use of this observation to develop a parallel pivoting strategy [8, procedure "*e-tree*", p. 233], which we describe in our terminology below.

ALGORITHM 2.1. *Parallel_Elimination* ($G$) { $G$ chordal graph }

```
begin
    $G_0 := G$;
    $i := 0$;
    while $G_i \neq \emptyset$ do begin
        $S_i :=$ the set of all nodes with no deficiency in $G_i$;
        $R_i :=$ a maximum independent subset of $S_i$;
        $G_{i+1} := G_i - R_i$ {eliminate the nodes of $R_i$ from $G_i$ };
        $i := i + 1$
    end
end.
```

It should be clear that we can exit from the "**while**" loop whenever the subgraph $G_i$ becomes a clique. In such a case, each subsequent $R_j$ (for $j \geqq i$) consists of only one node and the remaining nodes can be numbered in any order. Associated with the algorithm is a sequence of chordal graphs:

$$G_0, G_1, \cdots, G_m,$$

and a sequence of subsets of independent nodes:

$$R_0, R_1, \cdots, R_m,$$

where each $R_i$ is a maximum independent set of nodes with no deficiency in $G_i$.

It is worthwhile to point out that Jess and Kees [8] make the important observation that the set $S_i$ consists of disjoint cliques. This implies that any maximal independent subset of $S_i$ is maximum. Therefore it is sufficient to consider maximal independent subset $R_i$ of $S_i$. Intuitively, the algorithm eliminates as many nodes in "parallel" as possible in each step. This explains the appropriateness of the scheme for parallel factorization.

We include an example in Fig. 2.1 with eight nodes. On applying Algorithm 2.1, the following shows one possible choice of independent subsets:

| Step $i$ | $S_i$ | Selected $R_i$ |
|---|---|---|
| 0 | $\{a,b,c,f,h\}$ | $\{a,c,f,h\}$ |
| 1 | $\{b,d,g\}$ | $\{b,d\}$ |
| 2 | $\{e,g\}$ | $\{g\}$ |
| 3 | $\{e\}$ | $\{e\}$ |

The sequence of chordal graphs $\{G_i\}$ is also given in Fig. 2.1 to illustrate the algorithm.

Implicit in Algorithm 2.1 is that nodes are reordered in the order as given by the sequence $\{R_i\}$. Within each subset $R_i$, the nodes can be numbered in any order. Note that the resulting reordering from this algorithm may not be unique due to the different choices of the independent subset $R_i$ at each step, and the freedom to number nodes within $R_i$.

The description of Algorithm 2.1, as it is, does not offer an efficient implementation. Indeed, Jess and Kees [8, p. 238] point out that in this reordering algorithm, "the major source of complexity is the test on zero deficiency." But they have not addressed this zero-deficiency test problem. Our contribution in this paper is to provide a linear-time algorithm to determine a parallel pivoting sequence as specified by Algorithm 2.1.

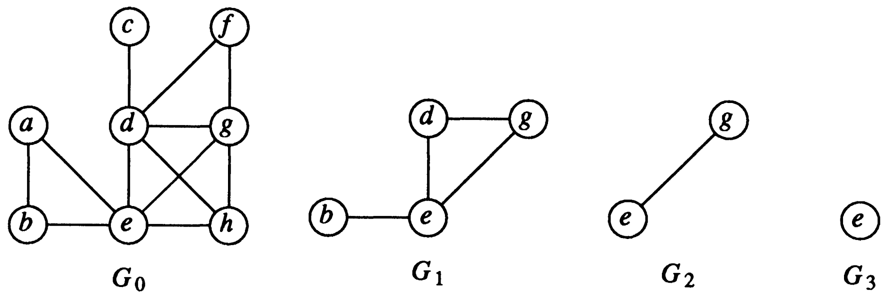


FIG. 2.1. *Sequence of chordal subgraphs by Algorithm* 2.1.

**2.2. Use of the parallel elimination reordering for nonchordal graphs.** Although Algorithm 2.1 is designed for chordal graphs, it is actually applicable in a more practical setting to parallel sparse Cholesky factorization. Consider a given sparse matrix $A$ with graph $G(A)$. Although the graph $G(A)$ is most likely nonchordal, it is well known that its filled graph is chordal [11]. Indeed, the ordering inherited from $A$ will clearly create no additional fill on the filled matrix, and hence is a perfect elimination ordering. The parallel elimination ordering of Algorithm 2.1 is, therefore, applicable to the filled graph of $A$.

Ideally, we want to find an ordering for the matrix $A$, which is appropriate for both reducing fill and exploiting parallelism. A modular approach to this is to determine first a good fill-reducing ordering, which defines the filled graph. Then, a reordering for this filled graph can be obtained by Algorithm 2.1, and the reordering minimizes the number of parallel elimination steps among all perfect elimination orderings of the filled graph. In other words, the competing issues of fill reduction and parallelism exploitation are being dealt with in separate steps. We may view this modular approach as having the following phases.

(a) (*Ordering*) Determine a fill-reducing ordering $P$ for the given $G(A)$;

(b) (*Filled Graph*) Form the filled graph of $G(PAP^t)$ explicitly or implicitly;

(c) (*Reordering*) Apply Algorithm 2.1 to the filled graph (which is chordal) to obtain an equivalent reordering $\tilde{P}$ of $G(A)$.

It should be pointed out that the problem of finding good orderings for reducing fill and exploiting parallelism is still under active research. The modular approach discussed here has the advantage of being simple and efficient. An alternative approach is to find such orderings directly from the structure of $G(A)$. The work in [1], [2] explores this direct approach.

**3. Zero-deficiency test.** The key to Algorithm 2.1 is the step that determines a maximum independent subset $R_i$ of nodes with no deficiency in the current chordal subgraph $G_i$. In this section, we provide a linear-time algorithm to find such an independent subset based on an efficient zero-deficiency test. We first provide a characterization of nodes with zero deficiency using a perfect ordering of the given chordal graph.

Let $G = G(A)$ be the given chordal graph and $x_1, x_2, \cdots, x_n$ be a perfect elimination node sequence. We shall use the notation $Adj_G(x_j)$ for the adjacent set of the node $x_j$ in $G$. Following Rose [11], we define the *monotone adjacent set* of the node $x_j$ to be

$$Madj_G(x_j) = \{x_i \in Adj_G(x_j) \mid i > j\}.$$

We shall refer to $deg_G(x_j) = |Adj_G(x_j)|$ and $Mdeg_G(x_j) = |Madj_G(x_j)|$ as the *degree* and *monotone degree* of $x_j$, respectively. When the chordal graph $G$ is clear from context, we shall omit the subscript $G$ and use $Adj(x_j)$, $Madj(x_j)$, $deg(x_j)$, and $Mdeg(x_j)$ instead.

For each node $x_j$, define $f_j$ as

$$f_j = \min\{k \mid x_k \in Adj(x_j) \cup \{x_j\}\}.$$

In matrix terms, $f_j$ is the first nonzero (including the diagonal) in the $j$th row of the sparse matrix $A$.

LEMMA 3.1. *$Madj(x_{f_j}) \cup \{x_{f_j}\} \subseteq Adj(x_j) \cup \{x_j\}$, for $j = 1, \cdots, n$.*

*Proof.* Let $k = f_j$. Since the sequence is a perfect elimination sequence, when $x_k$ is eliminated, $Madj(x_k)$ is the set of its adjacent nodes at the elimination time and is therefore a clique. But $x_j \in Madj(x_k)$ so that $x_j$ is adjacent to every node in $Madj(x_k) - \{x_j\}$. □

LEMMA 3.2. *The node $x_j$ has no deficiency in the chordal graph $G$ if and only if $Adj(x_j) \cup \{x_j\} \subseteq Madj(x_{f_j}) \cup \{x_{f_j}\}$.*

*Proof.* "*only if*" *part*. Let $k = f_j$. Assume that the node $x_j$ has no deficiency in the chordal graph $G$. Since $x_k \in Adj(x_j) \cup \{x_j\}$, and $Adj(x_j) \cup \{x_j\}$ forms a clique in $G$, we have

$$Adj(x_j) \cup \{x_j\} \subseteq Adj(x_k) \cup \{x_k\}.$$

The result then follows from the fact that $x_k$ is the first adjacent node of $x_j$ in the perfect ordering.

"*if*" *part*. Assume that the condition on adjacent sets of $x_j$ and $x_{f_j}$ is given. Again, let $k = f_j$. Since $Madj(x_k)$ is the set of adjacent nodes of $x_k$ at its elimination time, it, together with the node $x_k$, forms a clique in $G$. This implies that its subset $Adj(x_j) \cup \{x_j\}$ must also be a clique in $G$. □

COROLLARY 3.3. *If $f_j = j$, then the node $x_j$ has no deficiency in the chordal graph $G$.*

COROLLARY 3.4. *The node $x_j$ has no deficiency in the chordal graph $G$ if and only if $Adj(x_j) \cup \{x_j\} = Madj(x_{f_j}) \cup \{x_{f_j}\}$.*

Corollary 3.4 provides an efficient zero-deficiency test. By definition, a node $v$ has no deficiency if and only if

$$Adj(v) \cup \{v\} \subseteq Adj(w) \cup \{w\},$$

for every node $w$ adjacent to $v$. Corollary 3.4 makes use of a perfect elimination sequence to perform the deficiency test using (at most) one adjacent node of $v$. An even simpler test is provided in the next theorem based only on node degrees.

THEOREM 3.5. *The node $x_j$ has no deficiency in the chordal graph $G$ if and only if $deg(x_j) = Mdeg(x_{f_j})$.*

*Proof.* The "*only if*" part follows directly from Corollary 3.4. The "*if*" part follows from the results of Lemma 3.1 and Corollary 3.4. □

For a given chordal graph $G$ with a perfect elimination ordering, we can use Theorem 3.5 to determine the set of all nodes with no deficiency. The following algorithm adapts this idea to find a maximum independent subset $R$ of nodes with no deficiency. As before, we assume that $x_1, x_2, \cdots, x_n$ is a given perfect elimination sequence for $G$.

ALGORITHM 3.1. *No_Deficiency* $(G, R)$

```
begin
    $R := \emptyset$;
    for $j := 1$ to $n$ do
        compute $deg(x_j)$, $Mdeg(x_j)$, and $f_j$;
    unmark all nodes in $G$ ;
    for $j := 1$ to $n$ do
        if $x_j$ is unmarked and $deg(x_j) = Mdeg(x_{f_j})$
            then add $x_j$ to $R$ and mark nodes in $Adj(x_j)$
end.
```

The marking involved in the algorithm is to ensure that the set $R$ is independent. It is interesting to note that in Algorithm 3.1, if we omit the zero-deficiency test on degrees, it is essentially the algorithm by Gavril [4] to determine a maximum independent set (not necessarily of no deficiency) of a chordal graph. The linear time complexity of Algorithm 3.1 is clear and we state the following result without proof.

THEOREM 3.6. *Algorithm 3.1 finds a maximum independent subset of nodes with no deficiency in $G$ in $O(n + e)$ time, where $n$ is the number of nodes and $e$ the number of edges in the given chordal graph.*

**4. A linear reordering algorithm.** It is easy to incorporate Algorithm 3.1 ("*No_Deficiency*") into the basic parallel elimination algorithm of § 2. The "**while**" loop in Algorithm 2.1 can now be replaced by the following:

```
while $G_i \neq \emptyset$ do begin
    No_Deficiency $(G_i, R_i)$;
    $G_{i+1} := G_i - R_i$;
    $i := i + 1$
end
```

Unfortunately, this simple replacement does not make the time complexity of the overall parallel pivoting scheme linear. We need some more fine-tuning of the algorithm.

Recall in Algorithm 2.1, at step $i$, $G_i$ is the chordal graph under consideration. We use the notation $S_i$ to represent the set of all nodes with no deficiency in $G_i$. Since $R_i$ contains independent nodes, if $v \in R_i$, the adjacent nodes of $v$ can be excluded from $R_i$.

The next observation can be used to further discard nodes for membership consideration in $R_i$. It says that we only need to look at the adjacent nodes of $R_{i-1}$ in the search for the next independent set $R_i$.

THEOREM 4.1 [8]. *For $i > 0$, $R_i \subseteq S_i \subseteq Adj(R_{i-1}) \cap G_i$.*

COROLLARY 4.2. *For $i > 0$, $|R_i| \leqq |R_{i-1}|$.*

*Proof.* Consider a node $x$ in $R_{i-1}$. Since $Adj(x) \cap G_i$ is a clique and nodes in $R_i$ are independent, by Theorem 4.1, at most one vertex from $Adj(x) \cap G_i$ can belong to $R_i$. Therefore, the number of nodes in $R_i$ cannot exceed that of $R_{i-1}$. □

Before we give the overall reordering algorithm, we first discuss the computation of $f_j$, a quantity required for the zero-deficiency test of node $x_j$. We need a fast way to compute and update $f_j$ for each $j$, since the value of $f_j$ may change as edges and nodes are being removed from the graph during the course of the algorithm. In order to overcome this problem, we presort each adjacency list in ascending order of the node indices according to the given perfect elimination ordering. All adjacency lists can be sorted in $O(n + e)$ time by a careful application of bucket sort. After having adjacency lists sorted, then for each node $x_j$ we can find the node $x_{f_j}$ in $O(1)$ time, since $x_{f_j}$ is either $x_j$ or the first node in the adjacency list of $x_j$, whichever has the smaller index. It should be added that in practice, the adjacency lists are often already in this desired ascending order (see, for example, [9]).

ALGORITHM 4.1. *Parallel_Elimination* $(G)$ $\{G$ chordal graph$\}$

```
begin
    determine a perfect elimination sequence $x_1, x_2, \cdots, x_n$;
    $S_0 := \varnothing$;
    for $j := 1$ to $n$ do begin
        sort $Adj(x_j)$ in ascending order;
        compute $deg(x_j)$ and $Mdeg(x_j)$;
        $mark(x_j) := 0$;
        if $deg(x_j) = Mdeg(x_{f_j})$ then add $x_j$ to $S_0$
    end;
    $i := 0$;
    $G_0 := G$;
    while $G_i \neq \varnothing$ do begin
        $R_i := \varnothing$;
        $S_{i+1} := \varnothing$;
        for each $x_k$ of $S_i$ do begin
            if $mark(x_k) = i$ then begin
                add $x_k$ to $R_i$;
                for each node $x_j \in Adj_{G_i}(x_k)$ in ascending order do begin
                    remove $x_k$ from $Adj_{G_i}(x_j)$ and update $f_j$ if necessary;
                    $deg(x_j) := deg(x_j) - 1$;
                    if $j < k$ then $Mdeg(x_j) := Mdeg(x_j) - 1$;
                    if $deg(x_j) = Mdeg(x_{f_j})$ and $mark(x_j) \leqq i$
                        then add $x_j$ to $S_{i+1}$ and set $mark(x_j) := i + 1$
                end
            end
        end;
        $G_{i+1} := G_i - R_i$;
        $i := i + 1$
    end
end.
```

At step $i$ of the algorithm, the main "while-loop" determines the sets $R_i$ and $S_{i+1}$ simultaneously, using $S_i$. It makes an implicit use of Theorem 4.1 in the determination of $S_{i+1}$. The marker vector $mark(*)$ is employed in the algorithm to ensure the correct selection of nodes in $R_i$ and $S_{i+1}$.

During the execution of step $i$, for each node $x_j$ in $G_i$, $mark(x_j) \in \{0, i, i+1\}$. $mark(x_j) = i$ means $x_j \in S_i - S_{i+1}$ (unless $i = 0$); and $mark(x_j) = i + 1$ implies $x_j \in S_{i+1}$. Furthermore, if $x_j$ is not zero deficient in $G_i$, then $mark(x_j) = 0$. A node $x_k$ in $S_i$ with $mark(x_k) > i$ is not included into $R_i$ since its marker value must be $i + 1$. This means $x_k$ was adjacent to some node $x_{k'}$ which was removed from the graph into $R_i$ at an earlier time of step $i$. At the end of this step, $S_{i+1}$ contains all nodes with no deficiency in the (chordal) subgraph $G_i - R_i$. Moreover, at the end of the algorithm, we have $mark(x_j) = i$ if and only if $x_j \in R_i$.

There are two places in the algorithm where the zero-deficiency test of the form $deg(x_j) = Mdeg(x_{f_j})$ is performed. It should be emphasized that at the point of testing, both $deg(x_j)$ and $Mdeg(x_{f_j})$ have their correct values. The correctness of $deg(x_j)$ is rather obvious. The correctness of $Mdeg(x_{f_j})$ follows from the fact that $f_j \leqq j$ and that the degree update "for-loop" goes through nodes in ascending order. This implies that $Mdeg(x_{f_j})$ has already been updated.

The time for each operation in the algorithm can be charged to either a node or an edge of the graph so that the total charge received by each node or edge is $O(1)$. This guarantees that the time complexity of the algorithm is $O(n + e)$. Indeed, it is clear that for each $x_k$ in $R_i$, the loop for degree update of nodes in $Adj_{G_i}(x_k)$ and membership consideration of $S_{i+1}$ can be done in time proportional to $|Adj_G(x_k)|$. Summing over all nodes, we have it bounded by $O(e)$. Next, consider the total time for membership selection of $R_i$ from $S_i$ for all $i$. By Theorem 4.1, this total time is bounded by

$$\begin{aligned}|S_0| + \sum_{i=1}^{m} |S_i| &\leqq n + \sum_{i=1}^{m} |Adj_G(R_{i-1})| \\ &\leqq n + \sum_{x} |Adj_G(x)| = O(n+e).\end{aligned}$$

Furthermore, since a perfect elimination ordering of a chordal graph can be obtained in linear time [6], [12], we have thus proved the following theorem.

THEOREM 4.3. *Given a chordal graph $G$ with $n$ nodes and $e$ edges, Algorithm* 4.1 *correctly computes a parallel elimination ordering of $G$ in $O(n + e)$ time.*

**5. Concluding remarks.** Algorithm 4.1 is a linear time reordering algorithm that generates a perfect ordering for a given chordal graph suitable for parallel elimination. Indeed, it may be regarded as an efficient implementation of the parallel pivoting strategy by Jess and Kees [8].

This reordering scheme can be used in the parallel factorization of sparse matrices (whose graphs are not necessarily chordal). In such a case, this algorithm should then be applied to the filled graph of the given sparse matrix. The resulting equivalent reordering will be appropriate for the original matrix in terms of both preserving sparsity and exploiting parallelism.

**Acknowledgment.** The authors thank Professor Alex Pothen for his comments.

## REFERENCES

[1] G. ALAGHBAND AND H. F. JORDAN, *Multiprocessor sparse L/U decomposition with controlled fill-in*, ICASE Report No. 85-48, NASA Langley Research Center, Hampton, VA, 1985.

[2] R. BETANCOURT, *Optimal ordering of sparse matrices for parallel triangulation*, Technical Report, Dept. of Electrical and Computer Engineering, San Diego State University, 1985.

[3] D. CALAHAN, *Parallel solution of sparse simultaneous linear equations*, in Proc. 11th Annu. Allerton Conf. Circuit and Syst. Theory, pp. 729–735, Oct. 1973.

[4] F. GAVRIL, *Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph*, SIAM J. Comput., 1 (1972), pp. 180–187.

[5] J. A. GEORGE AND J. W. H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[6] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[7] J. W. HUANG AND O. WING, *Optimal parallel triangulation of a sparse matrix*, IEEE Trans. Circuits and Systems, CAS-26 (1979), pp. 726–732.

[8] J. A. G. JESS AND H. G. M. KEES, *A data structure for parallel L/U decomposition*, IEEE Trans. Comput., C-31 (1982), pp. 231–239.

[9] J. W. H. LIU, *Reordering sparse matrices for parallel elimination*, Tech. Report CS-87-01, Dept. of Computer Science, York University, 1987. (Parallel Computing, to appear.)

[10] F. J. PETERS, *Parallel pivoting algorithms for sparse symmetric matrices*, Parallel Comput., 1 (1984), pp. 99–110.

[11] D. J. ROSE, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. Read, Ed., Academic Press, New York, 1972, pp. 183–217.

[12] R. E. TARJAN AND M. YANNAKAKIS, *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM J. Comput., 13 (1984), pp. 566–579.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 108–125, February 1989

# PARALLEL ALGORITHMS FOR ZERO-ONE SUPPLY-DEMAND PROBLEMS*

NOAM NISAN† AND DANNY SOROKER‡

**Abstract.** A technique that yields fast parallel algorithms for several zero-one supply-demand problems is presented. *NC* algorithms are given for the following related problems:

(1) Given a sequence of supplies $a_1, \cdots, a_n$ and demands $b_1, \cdots, b_m$, construct a zero-one flow pattern satisfying these constraints, where every supply vertex can send at most one unit of flow to each demand vertex.

(2) Given a sequence of positive and negative integers summing to zero, representing supplies and demands, respectively, construct a zero-one flow pattern so that the net flow out of (into) each vertex is its supply (demand), where every vertex can send at most one unit of flow to every other vertex.

(3) Construct a digraph without self-loops with specified in- and out-degrees.

The results are extended to the case where the input represents upper bounds on supplies and lower bounds on demands.

**Key words.** parallel computation, graph theory, network flow

**AMS(MOS) subject classifications.** 68Q15, 68R10, 90B10

## 1. Introduction.

Supply-demand problems are fundamental in combinatorial optimization [3], [10]. In one formulation of the problem, the input is a network in which each arc has a nonnegative capacity, and each vertex has a certain supply or demand (possibly zero). The task is to find a flow function, such that the flow through each arc is no more than its capacity and the difference between the flow into a vertex and out of it is equal to its supply (or demand). This problem is equivalent to the general max flow problem, and can, therefore, be solved efficiently sequentially [10], [14], [5], but probably has no efficient parallel solution, since it is *P*-complete [6]. There are, however, many interesting special cases of this problem the solutions of which do not require the full power of general max flow.

In this paper we are concerned with several such problems. The first problem we discuss is as follows: Given a sequence of supplies, $a_1, \cdots, a_n$, and demands, $b_1, \cdots, b_m$, construct a zero-one flow pattern satisfying these constraints, where every supply vertex can send at most one unit of flow to each demand vertex. Equivalently, we can state this problem as that of constructing a zero-one matrix, $M$, having $a_i$ ones in the $i$th row and $b_j$ ones in the $j$th column (for all $1 \leqq i \leqq n$, $1 \leqq j \leqq m$). We will refer to this problem as the *matrix construction problem*. $M$ is called a *realization* for the input $(\vec{a}, \vec{b})$. There is a simple sequential algorithm for constructing a realization if one exists [3], [4]: Select any row, assign its ones to the columns having largest column sums, and repeat this procedure in the reduced problem. If this procedure gets stuck (i.e., some column sum becomes negative), then no realization exists.

This algorithm, although easy to implement sequentially, seems very hard to parallelize. Thus it is natural to ask if there is a fast parallel algorithm for this problem. Two

* Received by the editors October 1, 1987; accepted for publication (in revised form) May 5, 1988.

† Computer Science Division, University of California, Berkeley, California 94720. This research was supported by the National Science Foundation under grant DCR-8411954 and by a grant from the Digital Equipment Corporation.

‡ Computer Science Division, University of California, Berkeley, California 94720. Present Address, IBM Almaden Research Center, San Jose, California 95120. This research was supported by the Defense Advanced Research Projects Agency (DOD) DARPA order 4871, monitored by the Space and Naval Warfare Systems Command under contract N00039-84-C-0089, and by the International Computer Science Institute, Berkeley, California.

remarks are relevant to this question. First, the problem can be solved by network flow techniques. Since the capacities are small (polynomial in the size of the flow network), there are *Random NC* algorithms for the problem by reduction to maximum matching [9], [13]. Second, there is a simple sequential method for *testing* whether an instance, $(\vec{a}, \vec{b})$ is realizable [3], [1]. It is based on partial sums of the sequences, and can be implemented in *NC* in a straightforward manner. However, this method does not yield a way of *constructing* a realization. This is another example of the apparent gap between search and decision problems in the parallel realm [8].

We present a deterministic *NC* algorithm for the matrix construction problem. Let $n$ and $m$ denote, respectively, the number of rows and columns of the realization matrix, and assume without loss of generality that $n \geqq m$. Our algorithm can be implemented to run in time $O(\log^4 n)$ using $O(n^2 \cdot m)$ processors on a CRCW PRAM, or, alternatively, in time $O(\log^3 n)$ using $O(n^4 \cdot m)$ processors on an EREW PRAM. In terms of the output matrix $M$ when $n = \Theta(m)$ the number of processors is $O(|M|^{1.5})$ and $O(|M|^{2.5})$, respectively (where $|M| = n \cdot m$).

The algorithm is based on a careful examination of the network flow formulation of the problem. It exploits the fact that there are only a polynomial number of cuts that need to be considered and that this set of potentially min cuts has a natural ordering associated with it.

The methodology we develop enables us to solve the following two related problems (with the same time and processor bounds):

(1) *The symmetric supply-demand problem*. Given a sequence of positive and negative integers summing to zero, representing supplies and demands, respectively, construct a zero-one flow pattern so that the net flow out of (into) each vertex is its supply (demand), where every vertex can send at most one unit of flow to every other vertex. Notice that this problem is quite different than the matrix construction problem, since it does not have a "bipartite" nature.

(2) *The digraph construction problem*. Construct a *simple* directed graph with specified in- and out-degrees. This corresponds to constructing a zero-one matrix with specified row and column sums, where the diagonal entries are forced to be zero. References [3] and [1] give a simple sequential algorithm when the in- and out-degrees are sorted in the same order (i.e., a vertex with higher in-degree has higher out-degree). Our algorithm is the only one we know of for general orders that does not use max flow.

We extend our results to the case where the input represents upper bounds on supplies and lower bounds on demands.

An outline of the paper follows. In § 2 we explain our methodology in detail. We then state the matrix construction algorithm formally and finally discuss its parallel complexity (time and processor bounds). In § 3 we describe how our techniques can be used to yield a solution to the symmetric supply-demand problem. Section 4 contains a description of the algorithm for the digraph construction problem. Finally, in § 5 we describe our method for solving the supply-demand problems when we are given upper bounds on supplies and lower bounds on demands.

First, a few words about parallel algorithms. Our algorithms use, in various places, parallel prefix computations. An example of a problem in this category is as follows. Given a sequence $x_1, \cdots, x_n$, compute all sums of the form $\sum_{i=1}^{k} x_i$. Parallel prefix computation has been extensively studied in the literature (e.g., [2], [12]) and we will not discuss it in this paper, other than mentioning that it can be solved efficiently in parallel. Several other tools that we use implicitly are finding connected components [16] and various algorithms on trees [17].

## 2. The matrix construction problem.

**2.1. The slack matrix.** Our parallel algorithm is based on a careful analysis of the network flow formulation of the problem. The main tool we use is what we call the *slack matrix* which is similar to the "structure matrix" of Ryser [15]. In order to define the slack matrix, we need to look at the solution to our problem by network flow. Given the input $(\vec{a}, \vec{b})$: $a_1 \geqq a_2 \geqq \cdots \geqq a_n$, $b_1 \geqq b_2 \geqq \cdots \geqq b_m$, we construct a flow network $N$, as shown in Fig. 2.1. The vertex set consists of a source $s$; a sink $t$; and vertices $u_i$, $1 \leqq i \leqq n$, corresponding to rows and vertices $v_j$, $1 \leqq j \leqq m$, corresponding to columns. The arc set contains three types of arcs: for all $1 \leqq i \leqq n$, $1 \leqq j \leqq m$ there are arcs $(s, u_i)$ of capacity $a_i$; $(v_j, t)$ of capacity $b_j$; and $(u_i, v_j)$ of capacity one.

Let $S = \sum_{i=1}^{n} a_i = \sum_{j=1}^{m} b_j$. Clearly the max flow value in $N$ is bounded by $S$. Furthermore, a flow that satisfies all row and column sums is of value $S$. It follows (by the max flow-min cut theorem) that the problem instance $(\vec{a}, \vec{b})$ is realizable if and only if every directed cut in $N$ has capacity at least $S$.

Let $C = (C^s{:}C^t)$ be a directed cut in $N$ (i.e., the vertices are partitioned into two sets, $C^s$, $C^t$ subject to $s \in C^s$, $t \in C^t$). Say $C^s$ contains $x$ vertices from the set $\{u_1, \cdots, u_n\}$ and $m - y$ vertices from $\{v_1, \cdots, v_m\}$. Observe that if we replace $u_j$ by $u_i$ in $C^s$, for some $i < j$, then the capacity of the cut can only decrease. Similarly, replacing $v_k$ by $v_l$ in $C^s$ can only decrease the capacity of the cut, for $l > k$. It follows that the capacity of $C$ is no less than the capacity of the cut $C_{x,y}$, where $C^s_{x,y} = \{s\} \cup \{u_1, \cdots, u_x\} \cup \{v_{y+1}, \cdots, v_m\}$. Thus there are only $n \cdot m$ cuts,

$$\{C_{x,y} \mid 1 \leqq x \leqq n, 1 \leqq y \leqq m\},$$

which are potential min cuts. The cut $C_{x,y}$ is shown in Fig. 2.2. Therefore, necessary and sufficient conditions for the instance $(\vec{a}, \vec{b})$ to be realizable are that for every $1 \leqq x \leqq n$, $1 \leqq y \leqq m$:

$$\text{capacity}(C_{x,y}) = \sum_{i=x+1}^{n} a_i + \sum_{j=y+1}^{m} b_j + x \cdot y \geqq S$$

$$\Leftrightarrow \sum_{i=x+1}^{n} a_i + \left(S - \sum_{j=1}^{y} b_j\right) + x \cdot y \geqq S$$

$$\Leftrightarrow \sum_{i=x+1}^{n} a_i - \sum_{j=1}^{y} b_j + x \cdot y \geqq 0.$$

DEFINITION. The *slack* of $C_{x,y}$ of problem instance $(\vec{a}, \vec{b})$ is

$$sl_{\vec{a},\vec{b}}(x, y) = \sum_{i=x+1}^{n} a_i - \sum_{j=1}^{y} b_j + x \cdot y.$$

The *slack matrix* $SL_{\vec{a},\vec{b}}$ is the matrix whose $i,j$th entry is $sl_{\vec{a},\vec{b}}(i, j)$. For convenience we will use either the functional notation $(sl(x, y))$ or the matrix notation $(SL[x, y])$ when referring to slack.

PROPOSITION 2.1. *The instance $(\vec{a}, \vec{b})$ is realizable if and only if $SL_{\vec{a},\vec{b}}$ is non-negative.*

PROPOSITION 2.2. *Let $(\vec{a}, \vec{b})$ be an instance realizable by some matrix $M$ and assume that $sl_{\vec{a},\vec{b}}(x, y) = 0$. Then we have the following*:

(1) $M[i, j] = 1$ *for all* $1 \leqq i \leqq x$, $1 \leqq j \leqq y$;

(2) $M[i, j] = 0$ *for all* $x + 1 \leqq i \leqq n$, $y + 1 \leqq j \leqq m$.

*Proof*. Since $sl_{\vec{a},\vec{b}}(x, y) = 0$, the cut $C_{x,y}$ has capacity $S$, which means that in any max flow forward arcs (1) are all saturated, and backward arcs (2) all have zero flow. This situation is shown in Fig. 2.2. □

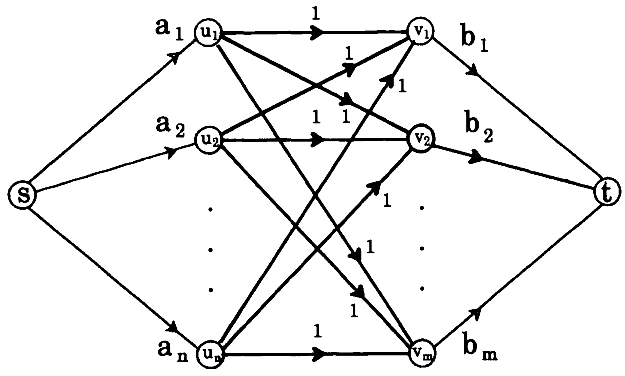


FIG. 2.1. *Flow network for solving the* 0-1 *matrix construction problem.*

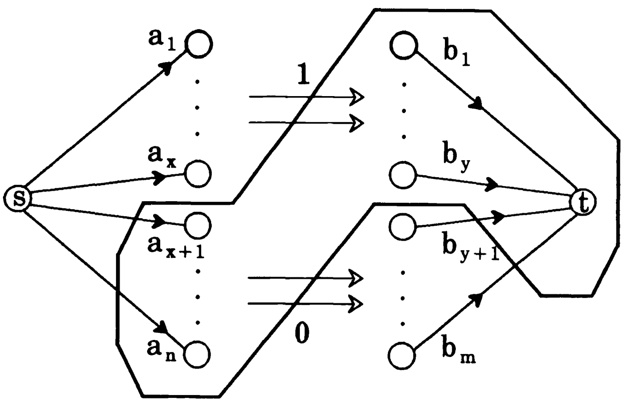


FIG. 2.2. *A tight cut—*$sl_{\vec{a},\vec{b}}(x, y) = 0$. *All forward arcs are saturated; all backward arcs have flow zero.*

If $sl_{\vec{a},\vec{b}}(x, y) = 0$, we will call $C_{x,y}$ a *tight cut*. Proposition 2.2 shows that existence of a tight cut simplifies the solution considerably. In fact it gives rise to the following divide and conquer approach. If $C_{x,y}$ is tight, constructing a matrix $M[1:n, 1:m]$ for the original problem is reduced to constructing the two submatrices, $M[x+1:n, 1:y]$ and $M[1:x, y+1:m]$. Of course, we are not always lucky enough to have a tight cut. Our approach is to *perturb* the input so as to improve our luck! Here is a high-level description of our algorithm:

(1) Perturb the inputs, $(\vec{a}, \vec{b})$. Call this new instance $(\vec{\alpha}, \vec{\beta})$.
(2) Recursively solve the instance $(\vec{\alpha}, \vec{\beta})$. Call the solution $M'$.
(3) Correct the matrix $M'$ to obtain a matrix, $M$, which solves the original instance, $(\vec{a}, \vec{b})$.

How do we perturb an instance? A *basic perturbation* can be viewed as shifting one unit from the poor to the rich in order to make the situation tighter: subtract one from $a_k$ and add one to $a_l$ for some $k > l$. We do not allow that a perturbation will change the ordering of the $a_i$'s, so it is necessary that $a_k > a_{k+1}$ and $a_l < a_{l-1}$ before the perturbation.

*Remark.* We will be discussing only perturbations of the row sums (the $a_i$'s). All this discussion holds for perturbation of the column sums as well.

PROPOSITION 2.3. *Let* $(\vec{a}, \vec{b})$ *be a problem instance, and let* $(\vec{\alpha}, \vec{\beta})$ *be obtained by shifting one unit from* $a_k$ *to* $a_l$ *for some* $k > l$. *Then* $sl_{\vec{\alpha},\vec{\beta}}(x, y) = sl_{\vec{a},\vec{b}}(x, y) - 1$ *if* $l \leq x < k$, *and* $sl_{\vec{\alpha},\vec{\beta}}(x, y) = sl_{\vec{a},\vec{b}}(x, y)$ *otherwise.*

*Proof*. This can be seen by looking at the formula for $sl$. □

This proposition shows that a basic perturbation reduces the slack of a certain set of cuts, and leaves the rest unchanged. This observation is the basis for our algorithm.

**2.2. One phase of perturbations.** Achieving poly-log recursion depth for the basic algorithm described in the previous section is a nontrivial matter. The reason is that it is hard to control which cut or cuts will become tight. Furthermore, since we have limited ourselves to perturbations that do not change the ordering of the $a_i$'s, it is not clear that a tight cut can always be obtained.

Say we are shifting units from $a_k$ to $a_l$ (for some $k > l$). How many units can we shift? Viewing the unit shifting as a sequential process (i.e., shifting one unit at each time step), we can shift until one of three things happens:

(1) $a_l$ becomes equal to $a_{l-1}$.

(2) $a_k$ becomes equal to $a_{k+1}$.

(3) $sl_{\vec{a},\vec{b}}(x, y)$ becomes zero, for some $l \leqq x < k$.

In case (3) progress is made, since a tight cut is created, and we can split the problem into two smaller problems. What about the first two cases? We observe that we have possibly *reduced the number of different* $a_i$ *values*. This observation is the key to our approach for performing perturbations.

DEFINITION. The *complexity of an instance* $(\vec{a}, \vec{b})$, denoted by comp $(\vec{a}, \vec{b})$, is the product of the number of different $a_i$ values and the number of different $b_j$ values.

Our parallel algorithm works in phases. The input to a perturbation phase is an instance of certain complexity, say $K$, and the output is one or more instances, each having complexity bounded by $c \cdot K$, for some constant $c < 1$. Finally, if the complexity of the input is less than a certain constant $B$, we construct a realization for it (this is the base case). Since the complexity of the input is initially bounded by $n \cdot m$, it follows that the total recursion depth is logarithmic in $n \cdot m$. We proceed to describe one perturbation phase. In this discussion we will derive the constants $c$ and $B$. For better exposition we will first describe a phase as a sequential process. The parallel implementation will be explained later.

In each phase either row sums or column sums are perturbed. The sequence that is perturbed (row or column sums) is that which has a larger number of different values. We will discuss a phase in which row sums are perturbed. Phases in which column sums are perturbed are essentially identical.

A phase starts by selecting a consecutive set of *active* rows, $\{h, h+1, \cdots, l\}$. The parameters $h$ and $l$ depend on the input, $(\vec{a}, \vec{b})$, and its complexity $K$, and will be derived later. Let $L = a_{l+1}$ and $H = a_{h-1}$. The perturbation is performed as follows. Repeatedly shift units from the lowest active row (initially row $l$) to the highest active row (initially row $h$). A row becomes inactive, and stops sending or receiving units, when its row sum either drops to $L$ or reaches $H$. The phase terminates when one of two things happens:

(1) At most one active row is left.

(2) $sl_{\vec{a},\vec{b}}(x, y)$ becomes zero, for some $h \leqq x < l$.

In case 1 no tight cuts have been obtained, but the row sums of all the active rows (except, possibly, one) have become either $L$ or $H$. Therefore the number of different row values decreases.

In case 2 one or more tight cuts are created, and the instance can be split, using Proposition 2.2, into two smaller instances ("smaller," in this case, means less rows *and* lower complexity).

Let $\alpha$, $\beta$, and $\gamma$ be the number of different values in the sets $\{a_1, \cdots, a_{h-1}\}$, $\{a_h, \cdots, a_l\}$ and $\{a_{l+1}, \cdots, a_n\}$, respectively. We want to select these parameters so as to minimize the complexity of the outputs of the following phase.

*Case* 1. The number of different row sums remaining is bounded by $\alpha + \gamma + 1$ (since the $\beta$ values corresponding to active rows disappeared, except for at most one).

*Case* 2. Zero slack is obtained for one or more rows in the range $[h, l-1]$. A simple calculation shows that the number of different row sums in the resulting instances is bounded either by $\alpha + \beta + 1$ or by $\beta + \gamma + 1$.

Thus we need to minimize the maximum of $\alpha + \beta + 1$, $\alpha + \gamma + 1$, and $\beta + \gamma + 1$ subject to $\alpha + \beta + \gamma = K$ (where $K = \operatorname{comp}(\vec{a}, \vec{b})$). The solution is, of course, to have $\alpha$, $\beta$, and $\gamma$ as equal as possible, i.e., all roughly $K/3$. From this calculation we can see that the complexity can be reduced by these perturbations as long as the number of different row values is more than five.

To summarize, if the input to a phase has complexity $K$, the outputs have complexity bounded by $\lceil 2K/3 \rceil + 1$. Thus the total number of phases is $O(\log(n \cdot m))$. The base case is any instance with at most five different row values and five different column values.

Next we discuss the parallel implementation of one perturbation phase. The first step is to calculate the new row sums and slack matrix under the assumption that none of the cuts become tight. If this new slack matrix is strictly positive then, indeed, we are in Case 1.

Let $p$ be the initial number of active rows $(p = l - h + 1)$. After the phase (assuming Case 1), there will be $q$ rows of value $H$, $p - q + 1$ rows of value $L$ and one row of value $I$, where $H > I \geqq L$. $q$ and $I$ are easy to calculate:

$$q = \left\lfloor \frac{\sum_{i=h}^{l}(a_i - L)}{H - L} \right\rfloor, \qquad I = \sum_{i=h}^{l} (a_i - L) \bmod (H - L).$$

Let $m_i = \min\{sl_{\vec{a},\vec{b}}(i, y) \mid 1 \leqq y \leqq m\}$, and let $m'_i$ be the new minimum slack in row $i$ after the phase is completed (assuming Case 1). Then we have the following:

$$\text{For } h \leqq i < h + q \quad m'_i = m_i - \sum_{j=h}^{i} (H - a_j);$$

$$\text{For } h + q \leqq i < l \quad m'_i = m_i - \sum_{j=i+1}^{l} (a_j - L).$$

If all the $m'_i$ are positive, then we are probably in Case 1. If not, we need to detect at what "timestep" (during the "sequential process") the first tight cut was created. This turns out to be a simple task for the following reason. If we plot the value of any entry in the slack matrix as a function of time, it decreases by one unit each step until some point in time, and remains constant from that point on. Thus the rows where the first zero slack occurs are the rows for which $m$ is minimum among the rows that have $m' \leqq 0$. The total number of units shifted in the phase is this minimum $m$ value. It is easy to compute the new row sums given the number of units shifted.

In Cases 1 and 2 we need to calculate the number of units shifted from row $j$ to row $i$, for every $h \leqq i < j \leqq l$. (These numbers will be used later, in the correction phase.) This calculation can be performed by a simple partial-sums computation.

**2.3. Correcting a perturbed solution.** After a realization is obtained for the perturbed instance we need to correct it in order to obtain a realization for the original instance. Clearly the required task is to shift units back to their original rows. The rows participating in the shifting of units are divided into two sets—the *donors* and the *receivers*, where donors shift units to the receivers during the perturbation phase, and get them back at the correction phase. Note that no row is both a donor and a receiver in any given phase.

Let $s(j, i)$ be the number of units shifted from the donor $j$ to the receiver $i$ in the perturbation phase.

DEFINITION. Let $M$ be a realization matrix. *Sliding* a unit from row $i$ to row $j$ means changing $M[i, k]$ from one to zero and $M[j, k]$ from zero to one, for some column $k$.

LEMMA 2.1. *Given any realization $M'$ of the perturbed instance it is always possible to correct it by sliding $s(j, i)$ units from receiver $i$ to donor $j$ for all receivers and donors.*

*Proof.* Again it is convenient to view the process of sliding units as a sequential one. Assume that some of the units have been slid, but less than $s(j, i)$ units have been slid from row $i$ to row $j$. Call the current matrix $M_1$. We will show that it is possible to slide a unit from row $i$ to row $j$ in $M_1$, which proves the lemma.

Since units were shifted from row $j$ to row $i$ in the perturbation phase, it is the case that $a_j$ was no larger than $a_i$ before the phase began. Other perturbations in which rows $i$ and $j$ might have participated only increased the row sum of $i$ and decreased the row sum of $j$. Now, since less than $s(j, i)$ units have been slid from row $i$ back to row $j$, it follows that row $i$ has more ones than row $j$ in $M_1$. By the pigeonhole principle there is some column $k$ such that $M_1[i, k] = 1$ and $M_1[j, k] = 0$. □

The implication of the proof above is that we do not need to be very careful in the way we slide units. The main problem we need to solve is that conflicts may arise when we slide many units in parallel. This could happen since a donor might have shifted units to many receivers, and a receiver might have received from many donors. Our goal is to break down the problem into a set of *independent* problems, which can all be solved in parallel. The first step is to get a formal description of the donor-receiver relation.

DEFINITION. The *donation graph* $G = (D, R, E)$ is a bipartite graph with a vertex, $d_j \in D$, representing each donor and a vertex, $r_i \in R$, representing each receiver, such that the edge $\{d_j, r_i\}$ is in $E$ if and only if $s(j, i) > 0$.

The following lemma plays a key role in simplifying the situation.

LEMMA 2.2. *The donation graph $G$ is a forest.*

*Proof.* Call a vertex *nontrivial* if its degree is greater than one. It follows from the way the perturbations were performed that each vertex $v$ has at most two nontrivial neighbors, one that became inactive before $v$, and one that became inactive after $v$. Furthermore, all the vertices can be ordered according to when they became inactive. Therefore $G$ cannot contain any cycles. □

We can see that a *matching* in the donation graph $G$ corresponds to an independent set of sliding problems. However, there is no guarantee that the edges of $G$ can be partitioned into a small set of matchings, since $G$ might have vertices of high degree. Thus a more subtle partition is required.

DEFINITION. A *constellation* is a subgraph of a given graph in which all connected components are *stars* (where a star is a tree with at most one nonleaf vertex).

LEMMA 2.3. *The edges of a forest can be partitioned into two* (*edge-disjoint*) *constellations.*

*Proof.* It suffices to show that the edges of a tree can be partitioned into two constellations. Let $T = (V, E)$ be a tree, and take it to be rooted at some vertex $P$. The *level* of a vertex is its distance from $P$. $v$ is the *parent* of $u$ if $\{u, v\} \in E$ and $v$ is closer to $P$ than $v$. The partition of $T$ into two constellations, $C_1 = (V, E_1)$, $C_2 = (V, E_2)$, is as follows:

$$E_1 = \{\{u, v\} \mid u \text{ is the parent of } v, \text{ the level of } u \text{ is even}\},$$

$$E_2 = \{\{u, v\} \mid u \text{ is the parent of } v, \text{ the level of } u \text{ is odd}\}.$$

An example of such a partition is shown in Fig. 2.3 □

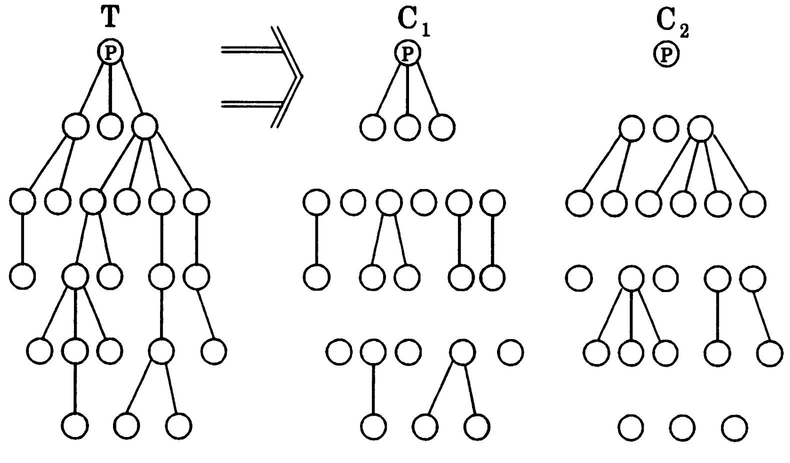


FIG. 2.3. *Partitioning a tree into two constellations.*

Our solution is based on the observation that a constellation corresponds to a set of independent sliding problems that we can solve in parallel. Therefore our approach will be to partition the donation graph into two constellations and then to slide units in two stages. The first stage corresponds to one constellation and the second to the other.

A star in the donation graph corresponds to several donors with a common receiver or several receivers with a common donor. These two cases are symmetric, so we will discuss only the first one. In what follows we describe a parallel algorithm that slides all the units corresponding to a star with receiver $R$ and donors $D_1, \cdots, D_d$. Let $M$ be a realization matrix of the perturbed instance we are about to correct. Let $r, d_1, \cdots, d_d$ denote the number of ones in rows $R, D_1, \cdots, D_d$, respectively, and let $s_i = s(D_i, R)$. We need to slide $s_i$ units from $R$ to $D_i$, for all $1 \leqq i \leqq d$ in parallel. Our approach is to solve a matching problem in the following bipartite graph, $B = (X, Y, E)$:

$$X = \{x_j \mid M[R,j] = 1\},$$

$$Y = \{y_{i,k} \mid 1 \leqq i \leqq d, 1 \leqq k \leqq s_i\},$$

$$E = \{\{x_j, y_{i,k}\} \mid M[D_i,j] = 0\}.$$

LEMMA 2.4. *Every matching of $B$ that covers all the vertices in $Y$ corresponds to sliding $s_i$ units from $R$ to $D_i$, for all $1 \leqq i \leqq d$ simultaneously.*

*Proof.* By construction, there are $\sum_{i=1}^{d} s_i$ vertices in $Y$, one corresponding to each unit that was shifted from some $D_i$ to $R$. There is an edge between $x_j$ and $y_{i,k}$ if and only if a unit can be slid from row $R$ to row $D_i$ in column $k$. The claim is, therefore, evident. □

At first sight it seems that we need to solve a maximum bipartite matching problem, but closer observation reveals the following lemma.

LEMMA 2.5. *Every <u>maximal</u> matching in $B$ is maximum.*

*Proof.* It suffices to show that any matching that does not cover all the vertices in $Y$ can be extended. The degree of $y_{i,k}$ in $B$ is, by definition, at least $r - d_i$. Before the perturbation phase the row sum of $R$ was no less than that of row $D_i$. After the perturbations, the row sum of $R$ increased by at least $\sum_{i=1}^{d} s_i$, and the row sum of $D_i$ decreased by at least one. Therefore,

$$\text{For all } i,k \quad \text{degree}\,(y_{i,k}) \geqq r - d_i \geqq \sum_{i=1}^{d} s_i + 1 = |Y| + 1.$$

Since any matching contains no more than $|Y|$ edges it follows that no partial matching is maximal. □

A maximal matching can be constructed efficiently in parallel [7], [11]. Our parallel algorithm is, therefore, the following. Construct the donation graph, and partition it into two edge-disjoint constellations $C_1$ and $C_2$. For each component of $C_1$ construct the bipartite graph $B$ as described, and find a maximal matching $F$ in it. For all edges of $B$ do the following in parallel. If $\{x_j, y_{i,k}\} \in F$, then slide a unit from $R$ to $D_i$ in column $j$. Finally, repeat this procedure on $C_2$ (with the updated matrix).

It follows from Lemmas 2.4 and 2.5 that after performing these operations all the perturbations (of the current phase) are corrected.

**2.4. The base case.** The base case for our algorithm is when the number of different values of row and column sums is bounded by a constant (five). The problem is then characterized by the different values: $a_1, \cdots, a_5$ and $b_1, \cdots, b_5$ and their multiplicities $n_1, \cdots, n_5$ and $m_1, \cdots, m_5$, respectively. Let $M$ be the realization matrix we construct, and let $M_{i,j}$ be the submatrix of $M$ induced on the rows with sum $a_i$ and columns with sum $b_j$. We construct $M$ in two steps.

*Step* 1: For each $i, j$, $1 \leqq i, j \leqq 5$, determine the number, $F_{i,j}$, of units in $M_{i,j}$.

*Step* 2: For each $i, j$, $1 \leqq i, j \leqq 5$, distribute the $F_{i,j}$ units between the different rows and columns of $M_{i,j}$.

We carry out step one by constructing a flow network of constant size, and finding a max flow in it. The network has twelve vertices: a source $s$, a sink $t$, five "row" vertices $u_1, \cdots, u_5$, and five "column" vertices $v_1, \cdots, v_5$. The arcs are of three kinds: arcs from $s$ to each $u_i$ with capacities $n_1 \cdot a_i$, from each $v_j$ to $t$ with capacities $m_j \cdot b_j$, and from each $u_i$ to each $v_j$ with capacities $n_i \cdot m_j$. This network is simply the result of taking the original network flow formulation for this problem, and compressing all "row" vertices with equal capacity into one vertex, and similarly for "column" vertices. Since this network is of constant size, a max flow can be constructed in constant time using standard sequential methods.

In Step two we convert the solution for the compressed network to a solution for the original network by distributing the flow along each compressed arc evenly between the arcs it defines. We do this by providing a solution for the following problem. Construct $M_{i,j}$ so that $x_{i,j}$ selected rows have each $r_{i,j}$ units, $y_{i,j}$ columns have each $c_{i,j}$ units and each of the remaining rows and columns have $r_{i,j} - 1$ and $c_{i,j} - 1$ units, respectively. First, it is not hard to see that

$$r_{i,j} = \left\lceil \frac{F_{i,j}+1}{n_i} \right\rceil, \qquad x_{i,j} = F_{i,j} \bmod n_i,$$

$$c_{i,j} = \left\lceil \frac{F_{i,j}+1}{m_j} \right\rceil, \qquad y_{i,j} = F_{i,j} \bmod m_j.$$

Assume we want each of the first $x_{i,j}$ rows and first $y_{i,j}$ columns to have $r_{i,j}$ and $c_{i,j}$ units, respectively. Our solution is to put the units of the first row in the first $r_{i,j}$ columns, the units of the second row in the cyclically next set of columns, etc. An example is shown in Fig. 2.4. A construction for arbitrary sets of selected rows and columns (not necessarily the first ones) is obtained from the one described above by simply permuting the rows and columns appropriately.

Now we are ready to construct a realization $M$ for the base case. The values $F_{i,j}$ determine the $x_{i,j}$ and $y_{i,j}$ values. All we need to ensure is that any two rows (columns)

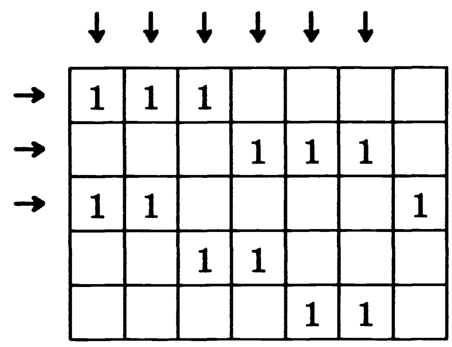


FIG. 2.4. *Structure of $M_{i,j}$ with five rows, seven columns, and 13 units. Selected rows and columns are marked with arrows.*

with equal row (column) sums get selected the same number of times. This can be done by selecting the first $x_{i,1}$ rows in $M_{i,1}$, the cyclically next set of $x_{i,2}$ rows in $M_{i,2}$ and so on, and similarly for columns.

Since $\sum_{j=1}^{5} F_{i,j} = n_i \cdot a_i$, the total number of rows selected in $\{M_{i,1}, \cdots, M_{i,5}\}$ is an integer multiple of $n_i$, and it follows that any two rows with equal row sums are selected the same number of times. A similar argument holds for columns. Thus the construction described yields a correct solution for the base case.

**2.5. The algorithm.** In this section we state the algorithm more formally. First, we give some notation. I.P is shorthand for "in parallel." Comments are between double parentheses. $l:k$ denotes a range of indices (in a matrix or a sequence). $\|$ denotes concatenation of sequences. $\#A$ is the cardinality of the set $A$.

```
procedure MATRIX_CONSTRUCTION ($\vec{a}, \vec{b}$)
((This is the recursive procedure for constructing a matrix $M$ with given row sums $\vec{a}$ and
column sums $\vec{b}$. The row and column sums are assumed to be given in a nondecreas-
ing order.))
    (1) Let $n$ = length of $\vec{a}$; $m$ = length of $\vec{b}$.
    (2) Compute $V_{\vec{a}}$ and $V_{\vec{b}}$—the number of different values in $\vec{a}$ and $\vec{b}$, re-
        spectively.
    (3) If $V_{\vec{a}} \leqq 5$ and $V_{\vec{b}} \leqq 5$ then return BASE_CASE ($\vec{a}, \vec{b}$).
    (4) ($\vec{\alpha}, \vec{\beta}, S, SL, pert, zerop$) = PERTURBATION ($\vec{a}, \vec{b}$).
    (5) If not zerop then $M'$ = MATRIX_CONSTRUCTION ($\vec{\alpha}, \vec{\beta}$).
    (6) Else let $x, y$ be such that $SL[x, y] = 0$ and either $a_x$ is in the middle third of
        the $\vec{a}$ values or $b_y$ is in the middle third of the $\vec{b}$ values. Do the following I.P:
        (6.1) I.P set $M'[i, j] = 1$ for all $1 \leqq i \leqq x$, $1 \leqq j \leqq y$.
        (6.2) I.P set $M'[i, j] = 0$ for all $x < i \leqq n$, $y < j \leqq m$.
        (6.3) $M'[x+1:n, 1:y]$ = MATRIX_CONSTRUCTION
                ($\vec{\alpha}[x+1:n], \vec{\beta}[1:y] - x$).
        (6.4) $M'[1:x, y+1:m]$ = MATRIX_CONSTRUCTION
                ($\vec{\alpha}[1:x] - y, \vec{\beta}[y+1:m]$).
    (7) $M$ = CORRECTION($M', S, pert$).
    (8) Return $M$.
end MATRIX_CONSTRUCTION
```

```
procedure PERTURBATION ($\vec{a}, \vec{b}$)
((This procedure computes one perturbation phase. The inputs are row sums $\vec{a}$ and
column sums $\vec{b}$. The outputs are new row and column sums $\vec{\alpha}$ and $\vec{\beta}$, respectively, the
slack matrix $SL$, the matrix of numbers of units shifted $S$, a variable pert indicating
whether row sums or column sums have been perturbed, and a variable zerop indicating
if zero slack is obtained.))
    (1) Let $n$ = length of $\vec{a}$; $m$ = length of $\vec{b}$.
    (2) Compute $V_{\vec{a}}$ and $V_{\vec{b}}$—the number of different values in $\vec{a}$ and $\vec{b}$, respec-
        tively. If $V_{\vec{a}} \geqq V_{\vec{b}}$ then set pert = "rows." Else set pert = "columns" and
        perform the rest of this routine with $\vec{b}$, $V_{\vec{b}}$ and $m$ instead of $\vec{a}$, $V_{\vec{a}}$ and $n$,
        respectively.
    (3) Find $h$ and $l$ for which $a_h \neq a_{h-1}$, $a_l \neq a_{l+1}$, and the number of different
        values in $\langle a_1, \cdots, a_{h-1} \rangle$ and $\langle a_h, \cdots, a_l \rangle$ are $\lfloor V_{\vec{a}}/3 \rfloor$ and $\lceil V_{\vec{a}}/3 \rceil$, respec-
        tively. Let $H = a_{h-1}$ and $L = a_{l+1}$.
    (4) Compute $q = \lfloor \sum_{i=h}^{l} (a_i - L)/(H - L) \rfloor$ and
        $I = \sum_{i=h}^{l} (a_i - L) \bmod (H - L)$.
    (5) Compute $SL[i,j]$ ((the slack matrix)) for all $1 \leqq i \leqq n$, $1 \leqq j \leqq m$ I.P.
    (6) Compute $m_i = \min \{SL[i,j] \mid 1 \leqq j \leqq m\}$ for all $h \leqq i \leqq l$ I.P.
    (7) Compute $m_i' = m_i - \sum_{j=h}^{i} (H - a_j)$ for all $h \leqq i < h + q$ I.P.
    (8) Compute $m_i' = m_i - \sum_{j=i+1}^{l} (a_j - L)$ for all $h + q \leqq i < l$ I.P.
    (9) If $m_i' > 0$ for all $h \leqq i < l$ then set $T = \sum_{i=h+q+1}^{l} (a_i - L) +$
        $\max \{0, a_{h+q} - I\}$. Else set $T = \min \{m_i \mid m_i' \leqq 0\}$, and set zerop to true.
   (10) Initialize $S[i,j] = 0$ for all $1 \leqq i, j \leqq n$.
   (11) $(\vec{\alpha}', S)$ = SHIFT_UNITS $(\langle a_h, \cdots, a_l \rangle, T, H, L)$.
   (12) Set $\vec{\alpha} = \langle a_1, \cdots, a_{h-1} \rangle \parallel \vec{\alpha}' \parallel \langle a_{l+1}, \cdots, a_n \rangle$.
   (13) Set $SL[i,j] = SL[i,j] - \sum_{k=h}^{i} \max \{0, \alpha_k - a_k\}$ for all $h \leqq i \leqq l$,
        $1 \leqq j \leqq m$ I.P.
   (16) Return $(\vec{\alpha}, \vec{b}, S, SL, pert, zerop)$.
end PERTURBATION
```

```
procedure SHIFT_UNITS ($\vec{a}, T, H, L$)
((Shifts a total of $T$ units between active rows with row sums $\vec{a}$. $H$ is the upper bound
on new rows sums and $L$ is the lower bound. Returns the new row sums and the matrix
$S$ of the numbers of units shifted between pairs of rows.))
    (1) Denote the elements of $\vec{a}$ by $a_h, \cdots, a_l$.
    (2) Compute for all $1 \leqq i \leqq T$ I.P:

        $d_i = \max \left\{ j \mid i \leqq \sum_{k=j}^{l} (a_k - L) \right\}$      ((donor of unit $i$))

        $r_i = \min \left\{ j \mid i \leqq \sum_{k=h}^{j} (H - a_k) \right\}$      ((receiver of unit $i$)).

    (3) Compute $S[i,j] = \#\{k \mid d_k = i, r_k = j\}$ for all $h \leqq j < i \leqq l$ I.P.
    (4) Compute $\alpha_i = a_i + r_i - d_i$ for all $h \leqq i \leqq l$ I.P.
    (5) Return $(\vec{\alpha}, S)$.
end SHIFT_UNITS
```

```
procedure CORRECTION ($M, S, pert$)
((This procedure computes one correction phase. The inputs are a realization matrix
$M$, a matrix $S$, containing amounts of units to be slid, and a variable pert indicating if
units need to be slid between rows or columns. The output is the matrix $M$ after it has
been corrected.))
    (1) Let $n$ = length of $S$.
    (2) Construct the donation graph, $G$ where
            $V(G) = \{1, \cdots, n\}$ $E(G) = \{\{i, j\} \mid S[i, j] > 0\}$
    (3) For every connected component $T$ of $G$ do I.P:
        (3.1) Partition $T$ into two constellations $C_1$ and $C_2$.
        (3.2) Perform SLIDE_UNITS $(C, M, S, pert)$ for every connected component
              $C$ of $C_1$ I.P.
        (3.3) Perform SLIDE_UNITS $(C, M, S, pert)$ for every connected component
              $C$ of $C_2$ I.P.
    (4) Return $M$.
end CORRECTION
```

```
procedure SLIDE_UNITS $(C, M, S, pert)$
((Units are slid in the matrix $M$, between one donor and many receivers or one receiver
and many donors. The vertices of the star $C$ are the participating rows/columns of $M$.
The matrix $S$ contains the numbers of units to be slid, and the variable pert indicates if
units need to be slid between rows or columns.))
    (1) Let $c$ be the unique nonleaf of $C$ ((If $C$ has exactly two vertices let $c$ be any
        one of them)). Let $l_1, \cdots, l_d$ be the remaining vertices of $C$.
    (2) If pert = "rows" then let $M_c, M_{l_1}, \cdots, M_{l_d}$ be rows $c, l_1, \cdots, l_d$ of $M$. Else
        let $M_c, M_{l_1}, \cdots, M_{l_d}$ be columns $c, l_1, \cdots, l_d$ of $M$.
    (3) If $S[c, l_1] > 0$ ((i.e., $c$ is a donor and $l_i$ are receivers)) then complement $M_c$,
        $M_{l_1}, \cdots, M_{l_d}$ I.P, and set comp to true. Let $s_i = \max\{S[l_i, c], S[c, l_i]\}$ ((the
        number of units to be slid from $M_c$ to $M_{l_i}$)) for $1 \leqq i \leqq d$.
    (4) Construct the bipartite graph, $B = (X, Y, E)$:
            $X = \{x_j \mid M_c[j] = 1\}$,
            $Y = \{y_{i,k} \mid 1 \leqq i \leqq d, 1 \leqq k \leqq s_i\}$,
            $E = \{\{x_j, y_{i,k}\} \mid M_{l_i}[j] = 0\}$.
    (5) Compute $F$, a maximal matching in $B$.
    (6) For all $\{x_j, y_{i,k}\} \in F$ do in parallel: set $M_c[j] = 0$ and $M_{l_i}[j] = 1$.
    (7) If comp then complement $M_c, M_{l_1}, \cdots, M_{l_d}$ I.P.
    (8) Copy $M_c, M_{l_1}, \cdots, M_{l_d}$ back into their original location in $M$ ((see step (2))).
end SLIDE_UNITS
```

```
procedure BASE_CASE $\vec{a}, \vec{b}$)
((Constructs a matrix $M$ with row sums $\vec{a}$ and column sums $\vec{b}$, where the number of
different values of elements in $\vec{a}$ and $\vec{b}$ is at most five.))
    (1) Let $a_1 > \cdots > a_k$ and $b_1 > \cdots > b_l$ be the values of the elements of $\vec{a}$ and
        $\vec{b}$, respectively, and let $n_1, \cdots, n_k$ and $m_1, \cdots, m_l$ be their respective multi-
        plicities.
    (2) Construct a flow network $N$ with vertices $s, t, u_1, \cdots, u_k, v_1, \cdots, v_l$ and the
        following arcs (for all $1 \leqq i \leqq k$, $1 \leqq j \leqq l$):
            from $s$ to $u_i$ with capacity $n_i \cdot a_i$,
            from $v_j$ to $t$ with capacity $m_j \cdot b_j$,
            from $u_i$ to $v_j$ with capacity $n_i \cdot m_j$.
```

```
(3) Find a max $s - t$ flow in $N$. For all $i, j$ let $F_{i,j}$ be the flow on the arc $(u_i, v_j)$.
(4) For all $i, j$ construct $M_{i,j}$ as shown in Fig. 2.4. There are $F_{i,j}$ mod $n_i$ selected
    rows, starting at row $(\sum_{h=1}^{j-1} F_{i,h} \bmod n_i) + 1$ (cyclically) and $F_{i,j}$ mod $m_j$
    selected columns, starting at column $(\sum_{h=1}^{i-1} F_{h,j} \bmod m_j) + 1$.
(5) Let $M$ be the appropriate concatenation of the $M_{i,j}$'s.
(6) Return $M$.
end BASE_CASE
```

**2.6. Parallel complexity.** The time and processor bounds of our algorithm depend on how we choose to implement the maximal matching routine. Two competing implementations are given in [7] and [11], respectively. On a graph with $e$ edges, Israeli and Shiloach's algorithm takes time $O(\log^3 e)$ and uses $O(e)$ processors on a CRCW PRAM. Luby's algorithm requires only $O(\log^2 e)$ time on an EREW PRAM, but uses $O(e^2)$ processors. It is straightforward, though somewhat tedious, to verify that all the other operations in one phase of *MATRIX_CONSTRUCTION* can be performed with the resources required for maximal matching (in both the implementations listed above).

There are $O(\log |M|)$ phases (as proven in § 2.2). In a correction phase for rows there are $O(n)$ parallel calls to maximal matching on bipartite graphs with $O(m^2)$ edges each. When columns are corrected, there are $O(m)$ calls, each of size $O(n^2)$. Thus the number of processors required is $O(nm(n+m)) = O(|M| \cdot (n+m))$ using [7], and $O(nm(n+m)^3) = O(|M| \cdot (n+m)^3)$ using [11]. When $n = \Theta(m)$ the processor requirements are $O(|M|^{1.5})$ and $O(|M|^{2.5})$, respectively.

**3. The symmetric supply-demand problem.** In this section we will show how the methodology developed in § 2 gives rise to a parallel algorithm to the symmetric problem. Here the input is a sequence of integers, $f_1 \geqq f_2 \geqq \cdots \geqq f_n$, summing to zero. The goal is to construct a flow pattern in which every vertex can send up to one unit of flow to any other vertex such that the flow out of $v_i$ minus the flow into it is $f_i$ (for all $1 \leqq i \leqq n$). The goal can be viewed as constructing an $n \times n$ zero-one matrix $M$ (where $M[i, j]$ is the amount of flow sent from vertex $i$ to vertex $j$) such that, for all $i$, the number of ones in row $i$ minus the number of ones in column $i$ is $f_i$. Note that changing the values along the main diagonal does not change the instance $M$ describes, so they can all be set to zero at the end of the computation.

Again we start with a network-flow formulation for the problem. The flow network has $n+2$ vertices: $s, t, v_1, \cdots, v_n$. If $f_i > 0$ then there is an arc from $s$ to $v_i$ with capacity $f_i$, and if $f_i < 0$ then there is an arc from $v_i$ to $t$ with capacity $f_i$. Also, there is an arc with capacity one from $v_i$ to $v_j$ for all $1 \leqq i, j \leqq n$. Examination of this network shows that there are only $n$ potential min cuts: of all cuts containing $x$ vertices with $s$, the one containing $v_1, \cdots, v_x$ is of smallest capacity. Thus, for this problem we have a *slack vector*. An analysis similar to the one in § 2 shows that, for all $1 \leqq x \leqq n$,

$$sl_{\bar{f}}(x) = x \cdot (n-x) - \sum_{i=1}^{x} f_i.$$

It is interesting to note that here, in contrast to the matrix construction problem, the object describing the slacks (a vector of length $n$) has a different size (and dimension) than the object being constructed (an $n \times n$ matrix).

A perturbation phase is performed in the same way as in § 2, except that there is only one sequence being perturbed (in contrast to separate row and column sequences). Again we have the property (similar to Proposition 2.3) that shifting a unit from $j$ to $i$ $(i < j)$ decreases the slacks at entries $i, i+1, \cdots, j-1$, and does not change the other entries.

A correction phase is, however, trickier than before. The reason is that if a unit is to be returned from entry $i$ to entry $j$, it can be done either by sliding a unit from row $i$ to row $j$ or by sliding a unit from column $j$ to column $i$. The equivalent of Lemma 2.1 holds here, but for each unit only one of the two ways of sliding listed above is guaranteed to exist. Furthermore, if we simultaneously try to slide units in rows and in columns, conflicts may arise.

Our solution is to perform the correction in two stages: first slide between rows, then slide between columns. The first stage is identical to a row-correction phase of § 2. The only difference is that the maximal matching computed does not necessarily cover all the vertices of one side of the bipartite graph $B$. After the first stage, we update the donation matrix (the $s(i, j)$'s), according to the numbers of units slid in the first stage. We then perform a column-correction phase for the resulting problem.

LEMMA 3.1. *Every maximal matching computed in the second stage is maximum.*

*Proof.* As in § 2.3, let $R, D_1, \cdots, D_d$ be the vertices of a star in the donation graph. Let $B_1 = (X_1, Y_1, E_1)$ be the bipartite graph for sliding between the rows corresponding to these vertices in the first stage. Let $B_2 = (X_2, Y_2, E_2)$ be the bipartite graph for sliding between the columns corresponding to these vertices in the second stage. First note that $Y_2$ is a subset of $Y_1$ (since $Y_1$ represents all the units that need to be slid in the correction phase and $Y_2$ represents the units that remain to be slid after the row-correction stage). Let $r$ ($d_i$, $1 \leqq i \leqq d$) denote the difference between the number of ones in row $R$ (respectively, $D_i$) and the number of ones in column $R$ (respectively, $D_i$).

Let $y_{i,k}$ be any vertex in $Y_2$. Let $\delta_1$ and $\delta_2$ denote its degrees in $B_1$ and $B_2$, respectively. By the same reasoning as in the proof of Lemma 2.5, $\delta_1 + \delta_2 \geqq |Y_1| + 1$. Let $q$ be the number of units slid in the row-correction stage. Since $q$ is the size of a maximal matching in $B_1$, it follows that $q \geqq \delta_1$. Also, by definition, $|Y_2| = |Y_1| - q$. Therefore $\delta_2 \geqq Y_2 + 1$, which proves the lemma. □

COROLLARY 3.1. *Every unit that is perturbed gets slid in one of the two stages.*

The base case is solved along the same lines described in § 2.4, but a few more details need to be handled. The base case is when there are at most five different values, $f_1 > \cdots > f_5$, with respective multiplicities $n_1, \cdots, n_5$. Again we start by finding a max flow in a constant size network (having seven vertices—$s, t, v_1, \cdots, v_5$) to determine the number of units $F_{i,j}$, in $M_{i,j}$. Now, contrary to the previous case, $\sum_{j=1}^{5} F_{i,j}$ need not be an integer multiple of $n_i$. Therefore, after distributing units evenly between all rows with the same $f$ value (as described in § 2.4), some of these rows will have $p$ units and some will have $p - 1$ units (for some appropriate $p$). Similarly, not all the columns with the same $f$ value will necessarily have the same number of units. We overcome this obstacle by observing that if $i$ and $j$ have the same $f$ value, and if row sum $i$ is greater by one than row sum $j$, then column sum $i$ should be greater by one than column sum $j$. Therefore, the problem is solved by (using terminology of § 2.4) selecting rows and columns in the same order.

Finally we note that the algorithm for the symmetric problem uses the same resources (time and number of processors) as the matrix construction algorithm (see § 2.6).

**4. Digraph construction.** In this section we describe our solution for the problem of constructing a simple digraph with specified in-degree and out-degree sequences. By "simple" we mean no self loops and no parallel arcs. Notice that if self loops are allowed, this problem is exactly the matrix construction problem described in § 2. The digraph construction problem can be stated as follows. Given two equal-length sequences, $(o_1, \cdots, o_n)$ and $(i_1, \cdots, i_n)$ (that are not necessarily sorted!), construct an $n \times n$ zero-one matrix, $M$, that has $o_k$ ones in row $k$ and $i_k$ ones in column $k$ (for all $1 \leqq k \leqq n$), so that all the elements on the main diagonal of $M$ are zero.

Our solution is based on the algorithm described in § 2. Again we start by looking at the network flow formulation for this problem. The network is almost identical to the one in Fig. 2.1, except that each vertex on the left is missing one outgoing arc, and each vertex on the right is missing one incoming arc. It is convenient to view the missing arcs as existing arcs with capacity zero. We will call these *blocked arcs* and the corresponding entries in the realization matrix *blocked entries*. Our first goal is to show that in this case too there are only $n^2$ potential minimum cuts. Let $a_1 \geqq \cdots \geqq a_n$ and $b_1 \geqq \cdots \geqq b_n$ be the sorted sequences of out-degrees and in-degrees, respectively (i.e., $\vec{a}$ is obtained by sorting $\vec{o}$ and $\vec{b}$ by sorting $\vec{i}$), and let $N$ be the network corresponding to $\vec{a}$ and $\vec{b}$ (similar to the one shown in Fig. 2.1). The capacity of the cut $C_{x,y}$ (as shown in Fig. 2.2) is, in this case

$$\text{capacity}\,(C_{x,y}) = \sum_{i=x+1}^{n} a_i + \sum_{j=y+1}^{n} b_j + x \cdot y - B(x,y)$$

where $B(x, y)$ is the number of blocked arcs crossing the cut. Since there is at most one blocked entry in every row and every column, a simple argument shows that if $a_x > a_{x+1}$ and $b_y > b_{y+1}$ then this cut has the smallest capacity among all cuts for which the $s$ side contains $x$ vertices on the left and $n - y$ vertices on the right. However, if, say, $a_x = a_{x+1}$ then the cut obtained by switching vertices $u_x$ and $u_{x+1}$ might have smaller capacity, since the number of blocked arcs crossing it could be greater by one. Therefore, if we want the cuts $C_{x,y}$ to be the only potential minimum cuts, we need to be careful about the ordering of "row" vertices corresponding to rows with equal row sums, and similarly for columns. The conditions we need to enforce on the order are, simply: if $a_x = a_{x+1}$, then the blocked entry in row $x$ should be in a lower-indexed column than the blocked entry in row $x + 1$. The symmetrical conditions should hold for columns.

These conditions can be obtained by two rounds of sorting: first sort rows according to row sums. Sort rows with equal sums according to the corresponding column sums (i.e., the correspondence given by the $\vec{o}$ and $\vec{i}$ sequences), breaking ties arbitrarily. Now, sort the columns according to column sums. Columns with equal sums are sorted according to the order of the corresponding rows that was obtained in the first round. No ties can arise, since there is, at this point, a total ordering of the rows.

After this preprocessing is done, we are ready to proceed along the same lines as the algorithm described in § 2, with a few modifications. The slack function is now

$$sl_{\vec{a},\vec{b}}(x,y) = \sum_{i=x+1}^{n} a_i - \sum_{j=1}^{y} b_j + x \cdot y - B(x,y).$$

By the discussion above, it is again true that an instance is realizable if and only if its slack matrix is nonnegative. If $sl_{\vec{a},\vec{b}}(x, y) = 0$ then $M[i, j] = 1$ for all $1 \leqq i \leqq x$, $1 \leqq j \leqq y$ *except for blocked entries*, and $M[i, j] = 0$ for all $x + 1 \leqq i \leqq n$, $y + 1 \leqq j \leqq n$.

The perturbation phases work identically here, since they only deal with the row and column sums, and not with the internal structure of the realization matrix.

In the correction phases there is a small modification—units should not be slid into blocked entries. This is fixed by modifying the bipartite graph $B$ in the obvious way. Also, we need to re-examine the proof of Lemma 2.5. It works out exactly right in this case, since it turns out that

$$\text{for all } i, k \quad \text{degree}\,(y_{i,k}) \geqq |Y|,$$

which is precisely sufficient (see the original proof).

The only tricky modification turns out to be for the base case. Again, there are at most five different row sum values and five different column sum values. The difficulty is that there are blocked entries scattered throughout. This spoils the simple cyclic realization that existed. We overcome this by partitioning the matrix into finer submatrices than in the previous case. Each of the $M_{i,j}$'s is partitioned further so that each submatrix either contains no blocked entries, or contains a blocked entry in *every* row and column.

Again we construct a realization in two steps. The first step is to determine the total number of units in each submatrix. This is done, here too, by solving a max flow problem (where the capacity of a submatrix is the number of nonblocked entries in it). Again, the network here is of constant size, so a max flow can be computed in constant time. In the second step, the units are distributed within the submatrices. The key here is to deal first with the submatrices containing blocked entries. It is not always possible to select arbitrary sets of rows and columns, but it *is* possible to distribute the units so that the discrepancy between any two rows or any two columns will be at most one unit. This can be done as follows. Say the blocked entries are along the main diagonal (this will always be the case because of the preprocessing), and let $k$ be the number of rows (and columns) of the submatrix. Let $d_r$ (the $r$th diagonal) be the set of entries $(i, j)$ for which $j - i \equiv r \pmod{k}$. If $F$ units are to be distributed, fill $d_1, \cdots, d_{\lfloor F/k \rfloor}$, and place the remaining units in $d_{\lfloor F/k \rfloor + 1}$. An example is shown in Fig. 4.1. Now, after the "problematic" submatrices have been dealt with, we can construct the submatrices with no blocked entries in the same fashion as described in § 2.4. The same arguments for proving validity of the scheme go through, because there is at most one blocked entry in every row or column.

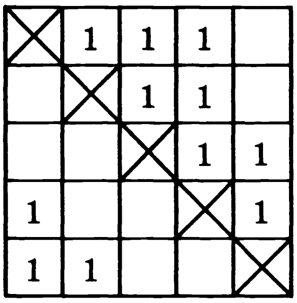


FIG. 4.1. *A* 5 × 5 *sub-matrix with blocked entries containing* 11 *units.*

**5. Bounds on supplies and demands.** Our parallel algorithm for the matrix construction problem can be extended to the case in which the sequences $\vec{a}$ and $\vec{b}$ represent upper bounds on row sums and lower bounds on column sums, respectively. This is a natural extension of the matrix construction problem when rows represent supplies and columns represent demands.

Let $U = \sum_{i=1}^{n} a_i$ and $L = \sum_{i=1}^{m} b_i$. Let $M$ be a realization matrix for the instance $(\vec{a}, \vec{b})$, and let $S$ be the number of ones in $M$. Then, clearly, $L \leqq S \leqq U$. Say we fix $S$. Then the problem condenses to the following. Modify the sequences $\vec{a}$ and $\vec{b}$ to obtain $\vec{\alpha}$ and $\vec{\beta}$, respectively, so that:

(1) $\alpha_i \leqq a_i$ and $b_j \leqq \beta_j$ for all $1 \leqq i \leqq n$, $1 \leqq j \leqq m$.

(2) $\sum_{i=1}^{n} \alpha_i = \sum_{j=1}^{m} \beta_j = S$.

(3) $(\vec{\alpha}, \vec{\beta})$ is realizable.

It is, of course, not always possible to satisfy all three conditions simultaneously. Thus our goal is find such a pair of sequences if it exists.

The key for obtaining the sequences $\vec{\alpha}$ and $\vec{\beta}$ is to consider the slack matrix, as defined in § 2.1. Recall that the condition for realizability is that all the slacks are nonnegative, and that

$$sl_{\vec{\alpha},\vec{\beta}}(x,y)=\sum_{i=x+1}^{n}\alpha_i-\sum_{j=1}^{y}\beta_j+x\cdot y$$

where $\alpha_1 \geqq \cdots \geqq \alpha_n$ and $\beta_1 \geqq \cdots \geqq \beta_m$.

LEMMA 5.1. *Let $a_1 \geqq \cdots \geqq a_n$ and $b_1 \geqq \cdots \geqq b_m$. Let $\vec{a}(k)(\vec{\beta}(l))$ be the sequence obtained from $\vec{a}(\vec{b})$ by subtracting one from $a_k$ (adding one to $b_l$). Then $(\vec{\alpha}(1), \vec{\beta}(m))$ is realizable if $(\vec{\alpha}(k), \vec{\beta}(l))$ is (for any $1 \leqq k \leqq n$, $1 \leqq l \leqq m$).*

*Proof.*

$$sl_{\vec{\alpha}(1),\vec{\beta}(m)}(x,y)-sl_{\vec{\alpha}(k),\vec{\beta}(l)}(x,y)=\sum_{i=x+1}^{n}(\alpha(1)_i-\alpha(k)_i)+\sum_{j=1}^{y}(\beta(l)_j-\beta(m)_j).$$

It is easy to see that for all values of $x$, $y$, $k$, and $l$ this difference is nonnegative, which proves the lemma. □

THEOREM 5.1. *Let $\vec{\alpha}_{(S)}$ be obtained from $\vec{a}$ by repeatedly subtracting one from the largest element $U-S$ times and let $\vec{\beta}_{(S)}$ be obtained from $\vec{b}$ by repeatedly adding one to the smallest element $S-L$ times. Then $(\vec{\alpha}_{(S)}, \vec{\beta}_{(S)})$ is realizable if there is any realizable pair of sequences $(\vec{\gamma}, \vec{\delta})$ where $\gamma_i \leqq a_i$, $\delta_j \geqq b_j$ (for all $1 \leqq i \leqq n$, $1 \leqq j \leqq m$) and $\sum_{i=1}^{n}\gamma_i=\sum_{j=1}^{m}\delta_j=S$.*

*Proof.* The proof is by induction on $U-S$, using Lemma 5.1. □

$(\vec{\alpha}_{(S)}, \vec{\beta}_{(S)})$ can be obtained from $(\vec{a}, \vec{b})$ efficiently in parallel by a simple partial-sums computation. The algorithm is as follows.

(1) For all $S$, $L \leqq S \leqq U$, do I.P:
   (1.1) Compute $\vec{\alpha}_{(S)}$ and $\vec{\beta}_{(S)}$.
   (1.2) Test if $(\vec{\alpha}_{(S)}, \vec{\beta}_{(S)})$ is realizable. This can be done by the following method described in [3, Chapt. 2, § 12]. $(\vec{\alpha}, \vec{\beta})$ (where $\alpha$ and $\beta$ are of lengths $n$ and $m$, respectively) is realizable if and only if, for $1 \leqq k \leqq m$:

$$\sum_{j=1}^{k}\beta_j \leqq \sum_{j=1}^{k}\alpha_j^*$$

   where $\alpha_j^* = \{i \mid a_i \geqq j\}$.
(2) Select an $S$ for which $(\vec{\alpha}_{(S)}, \vec{\beta}_{(S)})$ is realizable.
(3) Compute $M = MATRIX\_CONSTRUCTION\,(\vec{\alpha}_{(S)}, \vec{\beta}_{(S)})$.

Steps (1.1) and (1.2) are simple partial-sum computations, and can be implemented using $O(n+m)$ processors. Since steps (1) and (2) can be implemented within the time and processor bounds used for step (3), the algorithm has the same parallel complexity as the matrix construction algorithm. Note that we may perform step (2) with some criterion in mind (e.g., "construct a matrix with the smallest possible number of ones subject to . . .").

The extension of the symmetric supply-demand problem turns out to be even simpler. Here the natural extension would be that all the $f$ values represent upper bounds, since making a number "less positive" corresponds to less supply, and making a number "more negative" corresponds to more demand. So in an instance of this problems, the positive number would sum up to $+H$ and the negative number would sum up to $-L$, for some $H > L$.

Here, in contrast to the matrix construction problem, it is clear which value of $S$ works best (where $S$ is the sum of the positive entries, and minus the sum of the negative entries). By looking at the expressions for the slack vector, we can see that decreasing $S$ cannot ruin feasibility. Therefore $S$ should be selected to be as small as possible, i.e., $S = L$.

To summarize, only the positive $f$ entries should be modified. Again, as in the matrix construction problem, the best way to modify these numbers is to repeatedly subtract one unit from the largest entry until $H - L$ units have been subtracted.

**Acknowledgment.** We thank Richard Karp for suggesting the matrix construction and symmetric supply-demand problems and for interesting discussions.

## REFERENCES

[1] C. BERGE, *Graphs*, North Holland, Amsterdam, 1985.

[2] F. FICH, *New bounds for parallel prefix circuits*, in Proc. 15th Annual ACM Symposium on Theory of Computing, 1983, pp. 100–109.

[3] L. R. FORD AND D. R. FULKERSON, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[4] D. GALE, *A theorem on flows in networks*, Pacific J. Math, 7 (1957), pp. 1073–1082.

[5] A. GOLDBERG AND R. E. TARJAN, *A new approach to the maximum flow problem*, in Proc. 18th Annual ACM Symposium on Theory of Computing, 1986, pp. 136–146.

[6] L. M. GOLDSCHLAGER, R. A. SHAW AND J. STAPLES, *The maximum flow problem is logspace complete for P*, Theoret. Comput. Sci., 21 (1982), pp. 105–111.

[7] A. ISRAELI AND Y. SHILOACH, *An improved parallel algorithm for maximal matching in a graph*, Inform. Process. Lett., 22 (1986), pp. 57–60.

[8] R. M. KARP, E. UPFAL, AND A. WIGDERSON, *Are search and decision problems computationally equivalent?*, in Proc. 17th Annual ACM Symposium on Theory of Computing, 1985, pp. 464–475.

[9] ———, *Constructing a perfect matching is in random NC*, Combinatorica, 6 (1986), pp. 35–48.

[10] E. L. LAWLER, *Combinatorial optimization, Networks and Matroids*, Holt, Reinhart and Winston, New York, 1976.

[11] M. LUBY, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053.

[12] G. L. MILLER AND J. H. REIF, *Parallel tree contraction and its application*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, 1985, pp. 478–489.

[13] K. MULMULEY, U. V. VAZIRANI, AND V. V. VAZIRANI, *Matching is as easy as matrix inversion*, in Proc. 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 000–000.

[14] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[15] H. J. RYSER, *Traces of matrices of zeros and ones*, Canad. J. Math., 9 (1960), pp. 463–476.

[16] Y. SHILOACH AND U. VISHKIN, *An $O(\log n)$ parallel connectivity algorithm*, J. Algorithms, 3 (1982), pp. 57–67.

[17] R. E. TARJAN AND U. VISHKIN, *An efficient parallel biconnectivity algorithm*, SIAM J. Comput., 14 (1985), pp. 862–874.

SIAM J. DISC. MATH.
Vol. 2, No. 1, pp. 126–143, February 1989

# OPTIMAL ALGORITHMS FOR A PURSUIT-EVASION PROBLEM IN GRIDS*

KAZUO SUGIHARA† AND ICHIRO SUZUKI‡

**Abstract.** This paper discusses a problem of searching for and capturing a fugitive by a team of searchers in an $N \times N$ grid $G_N$ representing a system of *N avenues* and *N streets* in which a searcher can "see" a fugitive if and only if both are on the same avenue or the same street. A $0.5N^2 + O(N)$ time algorithm is presented for searching $G_N$ by a team of two searchers in order to decide whether there exists a fugitive in $G_N$. The algorithm is shown to be (1) optimal with respect to the number of searchers required for the case in which the fugitive can move at least as fast as the searchers, and (2) asymptotically optimal with respect to the worst case time complexity. An $O(N^2)$ time algorithm is also presented for capturing a fugitive in $G_N$ by a team of four searchers. The algorithm is shown to be asymptotically optimal with respect to the worst case time complexity.

**Key words.** graph search, grids, optimal algorithms, pursuit-evasion, pursuit game

**AMS(MOS) subject classifications.** 68Q20, 68Q25, 68R10

**1. Introduction.** The following pursuit-evasion problem has been known as the *game of cops and a robber*:

> Let $G$ be a finite connected graph. First, a number of cops are placed on some vertices of $G$. Then a robber is placed on some vertex of $G$. After that, the cops and the robber move alternately along the edges of $G$. The cops win if one of them occupies the same vertex as the robber (i.e., the robber is captured), and the robber wins otherwise.

It is assumed that the cops and the robber have complete information on the moves of their opponents. Relationships between the topology of a given graph and the number of cops needed to win the game has been investigated in [1]–[5], [9], [11], [13]. For example, Aigner and Fromme [1] showed that three cops always suffice to win if a given graph is planar.

In this paper we consider a similar problem in which complete information on the moves of the opponents is not available. In particular, we consider a pursuit-evasion problem in which *searchers* are required to search for and capture a *fugitive* in an $N \times N$ grid $G_N$ shown in Fig. 1. $G_N$ is viewed as a system of *N avenues* and *N streets* in which a searcher can "see" (and obtain the position of) the fugitive if and only if both are on the same avenue or the same street. In all algorithms presented except one, it is assumed that a searcher can communicate with other searchers and obtain their positions in real time irrespective of whether they are on the same avenue or the same street. We assume that searchers and a fugitive can move continuously and concurrently.

We first present a $0.5N^2 + O(N)$ time algorithm SEARCH1 for searching $G_N$ by a team of two searchers in order to decide whether there exists a fugitive in $G_N$. The algorithm SEARCH1 is shown to be (1) optimal with respect to the number of searchers for the case in which a fugitive can move at least as fast as the searchers, and (2) asymptotically optimal with respect to the worst case time complexity for the case in which the maximum speed of a fugitive is not zero (i.e., a fugitive is not immobile). The time

* Received by the editors October 26, 1987; accepted for publication (in revised form) August 4, 1988.

† Department of Information and Computer Sciences, University of Hawaii at Manoa, Honolulu, Hawaii 96822.

‡ Department of Electrical Engineering and Computer Science, University of Wisconsin, Milwaukee, Wisconsin 53201.

complexity of an algorithm is measured by the total time needed for the searchers to move.

As will be explained in § 3.3, SEARCH1 requires that the searchers must be able to communicate with each other in real time irrespective of their positions. In our second algorithm SEARCH2 a team of two searchers searches $G_N$, and it works even if the searchers can communicate only when they are on the same avenue or the same street. The time complexity of SEARCH2 is $N^2 + O(N)$. Both SEARCH1 and SEARCH2 work correctly irrespective of the speed of the fugitive.

Next we present an $O(N^2)$ time algorithm for capturing a fugitive in $G_N$ by a team of four searchers, under the assumption that the fugitive can move at most as fast as the searchers and the searchers can communicate with each other in real time irrespective of their positions. Here the fugitive is said to be captured if it occupies the same position as one of the searchers. The algorithm is shown to be asymptotically optimal with respect to the worst case time complexity for the case in which the fugitive is not immobile.

There is another problem similar to the problem of capturing a fugitive discussed in this paper. It is known as the *graph search* problem, which was first studied by Parsons [12] and further investigated by Kirousis and Papadimitriou [8] and Megiddo et al. [10]. This problem is different from ours in the following aspects:

(1) Searchers in the graph search problem are assumed to be "blind" in the sense that they can detect a fugitive only when they capture it.

(2) In the graph search problem, a fugitive can move with unbounded speed.

It is not difficult to show that $N+1$ searchers are necessary and sufficient for capturing a fugitive in $G_N$ under these assumptions.

The pursuit-evasion problem considered in this paper may have applications in motion coordination of multiple robots [6], [7], [14]–[17]. Basically, motion coordination is the problem of generating a plan for moving a number of robots from their initial positions to *given* final positions avoiding collision. A slightly different but equally interesting problem would be that of coordinating the motion of robots so that the robots as a team will achieve a given goal that may involve other moving objects, such as robots that are not members of the team. Achieving such goals can be more difficult than simply moving the robots to their final positions, since the possible movement of other objects whose behavior is not always predictable must also be considered. The problem we consider can be regarded as an example of problems of this type.

We note that the pursuit-evasion problem in $G_N$ considered in this paper can be regarded as a special case of more general problem of searching a two-dimensional region that contains obstacles of various sizes and shapes.

In § 2 we introduce some terminologies and assumptions. The results on searching $G_N$ are presented in § 3. Section 4 describes the algorithm for capturing a fugitive. Concluding remarks are found in § 5.

**2. Preliminaries.** An $N \times N$ *grid* $G_N$ ($N \geqq 2$) is the set of points $(x, y)$ in the plane such that (1) $0 \leqq x, y \leqq N-1$ and (2) at least one of $x$ and $y$ is an integer (see Fig. 1). $G_N$ can be viewed as the union of the following $2N$ line segments:

(a) The line segment between $(i, 0)$ and $(i, N-1)$, called *avenue* $i$ $(0 \leqq i \leqq N-1)$.

(b) The line segment between $(0, j)$ and $(N-1, j)$, called *street* $j$ $(0 \leqq j \leqq N-1)$.

For convenience of discussion, a point $(x, y)$ in $G_N$ is called a *vertex* if $x$ and $y$ are both integers. The line segment between two vertices $(x, y)$ and $(x', y')$ such that $|x-x'| + |y-y'| = 1$ is called an *edge*. We say, for example, that vertices $(5, 8)$ and $(7, 5)$ are to the *north* and to the *east* of vertex $(5, 5)$, respectively. *South* and *west* will also be used in the same manner. The distance between two points in $G_N$ is defined to be the length of a shortest path between them in $G_N$.

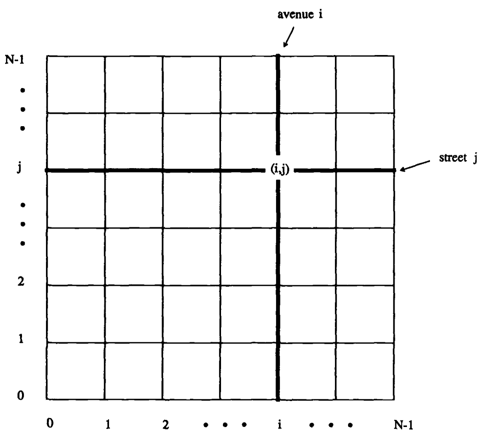


FIG. 1. $N \times N$ grid $G_N$.

Searchers and a fugitive are represented by their positions in $G_N$. Any number of searchers and a fugitive can occupy the same position at the same time. Throughout this paper we assume that a searcher can "see" (and obtain the position of) a fugitive if and only if both are on the same avenue or on the same street. We assume that searchers and a fugitive can move continuously and concurrently.

The pursuit-evasion problem in $G_N$ consists of two subproblems, searching and capturing. In the search problem, a team of searchers must decide whether there exists a fugitive hiding in $G_N$. In the capture problem, searchers must capture a fugitive, where the fugitive is said to be *captured* if it occupies the same position as one of the searchers. Algorithms for these subproblems should ensure that a given goal is achieved eventually irrespective of the movement of the fugitive.

Algorithms for the pursuit-evasion problem are analyzed in terms of the number of searchers and the time complexity. In this paper we assume that the time needed for a searcher to communicate with other searchers and compute its next move is negligible. Thus the time complexity of an algorithm is the total time needed to move the searchers.

## 3. Search problem.

In this section we consider the problem of searching $G_N$ by a team of searchers in order to decide whether there exists a fugitive in $G_N$. For convenience of discussion, we assume that there exists at most one fugitive in $G_N$, and we call it $Z$. The argument that follows remains valid even if there is more than one fugitive in $G_N$.

### 3.1. Algorithm SEARCH1.

In Algorithm SEARCH1 given below two searchers $A$ and $B$ search $G_N$. It is assumed that the searchers can communicate with each other in real time irrespective of their positions, although this assumption is not essential for solving the search problem, as will be discussed later. $A$ and $B$ traverse $G_N$ as described below, and when the traversal is completed, they conclude that there exists no fugitive in $G_N$ if and only if none of them has seen a fugitive during the traversal.

$G_N$ is traversed in $N-1$ phases, Phases 1 through $N-1$. Phase $k+1$ can be started only after Phase $k$ is completed. Phase $k$ ($k \geqq 2$) consists of subphases (a), (b), and (c), which are executed in this order. Phase 1 does not have subphases (a) and (b). Initially, $A$ and $B$ are in $(1, Y_1)$ and in $(0, Y_2)$, respectively, where $Y_1 = \lceil N/2 \rceil - 1$ and $Y_2 = N - \lceil N/2 \rceil$. When Phase $k$ is completed, $A$ and $B$ are in $(k, 0)$ and $(k-1, N-1)$, respectively, if $k$ is odd, and in $(k, Y_1)$ and $(k-1, Y_2)$, respectively, if $k$ is even. The traversal is illustrated in Fig. 2.

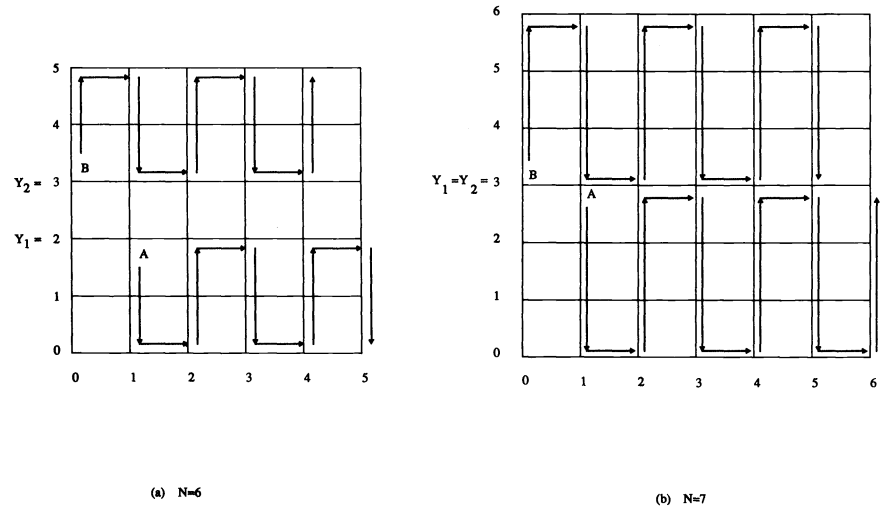


FIG. 2. *Traversal of* $G_N$ *determined by* SEARCH1.

ALGORITHM SEARCH1.

**Phase 1:** (c) $A$ moves south from $(1, Y_1)$ to $(1, 0)$, and $B$ moves north from $(0, Y_2)$ to $(0, N-1)$, in parallel.

**Phase $k$:** $(k = 2, 4, 6, \cdots)$

(a) $A$ stays in $(k-1, 0)$, and $B$ moves east from $(k-2, N-1)$ to $(k-1, N-1)$.

(b) $B$ stays in $(k-1, N-1)$, and $A$ moves east from $(k-1, 0)$ to $(k, 0)$.

(c) $A$ moves north from $(k, 0)$ to $(k, Y_1)$, and $B$ moves south from $(k-1, N-1)$ to $(k-1, Y_2)$, in parallel.

**Phase $k$:** $(k = 3, 5, 7, \cdots)$

(a) $A$ stays in $(k-1, Y_1)$, and $B$ moves east from $(k-2, Y_2)$ to $(k-1, Y_2)$.

(b) $B$ stays in $(k-1, Y_2)$, and $A$ moves east from $(k-1, Y_1)$ to $(k, Y_1)$.

(c) $A$ moves south from $(k, Y_1)$ to $(k, 0)$, and $B$ moves north from $(k-1, Y_2)$ to $(k-1, N-1)$, in parallel.

Note that SEARCH1 works only if searchers $A$ and $B$ can communicate with each other even if they are not on the same avenue or the same street. For example, $B$ can start executing subphase (a) of Phase 2 only after $A$ notifies $B$ that it has arrived at $(1, 0)$.

In order to analyze the time complexity of a search algorithm, we assume that the searchers move at a speed of one edge per unit time. (That is, they move over a distance of one (1) in unit time.) Then the time complexity is measured by the number of time units needed. In each phase of SEARCH1, subphases (a) and (b) can be executed in two time units. Subphase (c) can be executed in $Y_1$ $(=\lceil N/2 \rceil)$ time units, since searchers $A$ and $B$ move over $Y_1$ edges each, in parallel. Therefore, the time complexity of SEARCH1 is $Y_1 + (N-2)(2+Y_1) = 0.5N^2 + O(N)$.

**3.2. Optimality of SEARCH1.** Let $e$ be an edge of a grid $G_N$. We say that $e$ is *clear* at time $t$ if $A$ and $B$ "know" that there exists no fugitive in $e$. For example, $e$ is clear at

$t$ if (1) $A$ and $B$ have not seen a fugitive, and (2) $A$ and $B$ have moved in such a way that a fugitive cannot be in $e$ at $t$ unless it has already been detected by $A$ or $B$. Also, if a searcher is on avenue $i$ and there exists no fugitive on avenue $i$, then all edges on avenue $i$ are clear at that moment. (Note that $A$ and $B$ always have the same knowledge, since we are assuming that they can communicate in real time irrespective of their positions.)

For integers $i$ and $j$ ($0 \leqq i, j \leqq N-2$), the four vertices $(i, j)$, $(i+1, j)$, $(i, j+1)$, and $(i+1, j+1)$ together with the four edges between them are called *block* $B\ (i, j)$ (see Fig. 3). Block $B\ (i, j)$ is said to be *clear* at time $t$ if and only if all four edges of $B(i, j)$ are clear at time $t$. An edge or a block is said to be *contaminated* if it is not clear. Initially, all edges of $G_N$ except those which are seen by the searchers are contaminated, and hence all blocks except possibly one are contaminated.

THEOREM 1. SEARCH1 *correctly determines whether there exists a fugitive in* $G_N$, *even if the fugitive can move at any high speed.*

*Proof.* For each $k$ ($1 \leqq k \leqq N-1$), let CLEAR($k$) be the proposition that block $B(i, j)$ is clear for all $0 \leqq i \leqq k-1$ and $0 \leqq j \leqq N-2$. That is, CLEAR($k$) states that $A$ and $B$ know that there exists no fugitive in the subregion of $G_N$ consisting of points $(x, y)$ such that $0 \leqq x \leqq k$ and $0 \leqq y \leqq N-1$. In Phase 1, searchers $A$ and $B$ stay on avenues 1 and 0, respectively, so that they can see a fugitive that appears on these avenues. Furthermore, in Phase 1, $A$ and $B$ examine all edges between avenues 0 and 1. Therefore, when Phase 1 is completed, either CLEAR(1) holds or at least one of $A$ and $B$ has seen fugitive $Z$. Now assume that $A$ and $B$ did not see $Z$, and let us consider Phase 2. Since at least one of $A$ and $B$ stays on avenue 1 during subphases (a) and (b) of Phase 2, when subphase (b) is completed, either CLEAR(1) holds or at least one of $A$ and $B$ has seen $Z$. In subphase (c), $A$ and $B$ stay on avenues 2 and 1, respectively, and examine all edges between avenues 1 and 2. Therefore, when Phase 2 is completed, either CLEAR(2) holds or at least one of $A$ and $B$ has seen $Z$. By applying the same argument repeatedly, we can show that when Phase $N-1$ is completed, either CLEAR($N-1$) holds (i.e., $A$ and $B$ know that there is no fugitive in $G_N$) or at least one of $A$ and $B$ has seen $Z$. It is easy to see that the argument given above is valid irrespective of the speed of $Z$. □

Theorem 2 establishes the optimality of SEARCH1 with respect to the number of searchers for the case in which the fugitive can move at least as fast as the searchers.

THEOREM 2. $G_N$ *cannot be searched by a single searcher if the fugitive can move at least as fast as the searcher.*

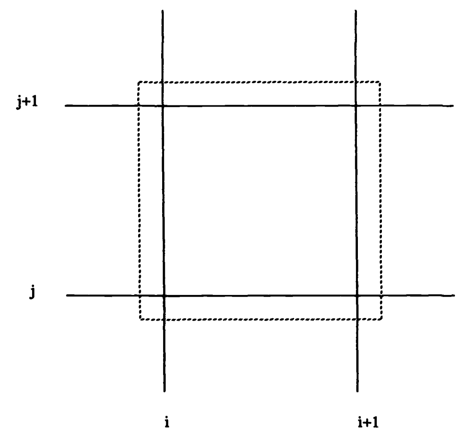


FIG. 3. *Block* $B(i, j)$.

*Proof.* Consider the case of $N = 2$. Since fugitive $Z$ can move at least as fast as searcher $A$, $Z$ can always be at the "opposite" position from $A$ around block $B$ (0, 0) (see Fig. 4). Therefore it is possible that $A$ will never see $Z$. Similar scenarios can be constructed for any $N > 2$. □

*Remark* 1. If the speed of fugitive $Z$ is sufficiently low compared to that of a searcher, then $G_N$ can be searched by a single searcher $A$. For example, if the speed of $Z$ is at most one $(2N - 1)$th of that of $A$, then $A$ can search $G_N$ in $N^2 - 1$ time units by traversing avenues 0, 1, $\cdots$, and $N - 1$ as shown in Fig. 5. Since $A$ moves at a speed of one edge per unit time, in this traversal every street is examined by $A$ repeatedly at intervals of at most $2N - 1$ time units. Since it takes at least $2N - 1$ time units for $Z$ to move from one avenue to another, $Z$ cannot change avenues without being detected by $A$. Thus $Z$ will eventually be detected, since $A$ examines all avenues. To the authors' knowledge, it is not known exactly when two searchers are required. □

Lemmas 1, 2, and 3 are used to derive Theorem 3, which establishes the optimality of SEARCH1 with respect to the time complexity for the case in which the maximum speed of the fugitive is not zero (i.e., the fugitive is not immobile) and the number of searchers is $O(1)$.

LEMMA 1. *If a search algorithm can search $G_N$ by time $T$, then there exists a search algorithm such that*

(1) *$G_N$ can be searched by time $T$;*

(2) *A searcher does not stop nor change directions of its movement in the middle of an edge; and*

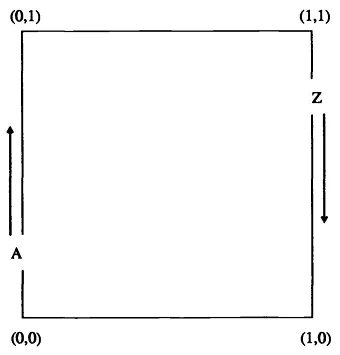


FIG. 4. *Illustration for Theorem 2.*

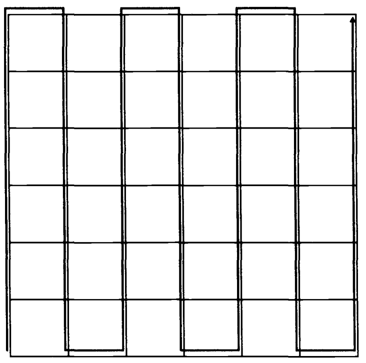


FIG. 5. *A $\theta(N^2)$ traversal by a single searcher.*

(3) *When a searcher moves from a vertex to an adjacent vertex, it moves at the maximum speed of one edge per unit time.*

*Proof.* Suppose that searcher $A$ moves from vertex $(i, j)$ to vertex $(i + 1, j)$ along street $j$ without visiting any other vertex. Assume that $A$ leaves $(i, j)$ at time $t$ and reaches $(i + 1, j)$ at time $t'$. If $A$ stops in the middle of the edge between the vertices or $A$ moves at a speed lower than the maximum of one edge per unit time, then we have $t' > t + 1$. Clearly, $A$ can only see street $j$ during the time interval $(t, t')$, and $A$ can see street $j$ and avenue $i + 1$ at time $t'$. On the other hand, if $A$ leaves $(i, j)$ at time $t$, moves toward $(i + 1, j)$ along street $j$ at a constant speed of one edge per unit time, reaches $(i + 1, j)$ at time $t + 1$, and stays in $(i + 1, j)$ from time $t + 1$ to $t'$, then $A$ obtains at least as much information as in the first case, since $A$ not only can see street $j$ during the time interval $(t, t']$, but also can see avenue $i + 1$ during $[t + 1, t']$. By using a similar argument, we can show that changing directions of movement in the middle of an edge does not make an algorithm faster. □

To prove a lower bound on the worst case time complexity of searching, in the following lemmas we only need to consider search algorithms that satisfy conditions (2) and (3) of Lemma 1.

LEMMA 2. *If a block becomes clear at time $t > 0$ for the first time after a search algorithm is started at time zero, then there exists some $\Delta > 0$ and an edge $e$ of the block such that*

(1) *$e$ is contaminated in $[t - \Delta, t)$; and*

(2) *At least one searcher can see $e$ at time $t$.*

*Proof.* If all edges of a block become clear simultaneously before time $t$, then the block is clear before $t$, contradicting the assumption. This, together with the assumption that the searchers traverse the edges at a constant speed in one direction, implies that there exist some $\Delta > 0$ and an edge $e$ of the block such that (1) $e$ is contaminated in $[t - \Delta, t)$, and (2) none of the edges that are clear at time $t - \Delta$ become contaminated in $[t - \Delta, t]$. Since a contaminated edge can become clear only when it is seen by a searcher, edge $e$ must be seen by a searcher at time $t$. □

LEMMA 3. *Suppose that the maximum speed of fugitive $Z$ is not zero. If block $B(i, j)$ becomes clear at time $t > 0$ for the first time after a search algorithm is started at time zero, then at time $t$ there exist a searcher in either avenue $i$ or $i + 1$ and a searcher in either street $j$ or $j + 1$.*

*Proof.* Let the four edges of block $B(i, j)$ be called $a$, $b$, $c$, and $d$, respectively, as shown in Fig. 6. Lemma 2 implies that at time $t$ a searcher must see an edge that was contaminated just before $t$. Therefore, at time $t$, there exists a searcher on one of avenue $i$, avenue $i + 1$, street $j$, and street $j + 1$. Without loss of generality, we can assume that for some $\Delta > 0$ edge $a$ is contaminated in $[t - \Delta, t)$ and searcher $A$ is on street $j + 1$ at time $t$ (see Fig. 6). Since nothing has to be proved if $A$ is in $(i, j + 1)$ or $(i + 1, j + 1)$ at time $t$, we assume that $A$ is in $(k, j + 1)$ at time $t$, where $k < i$ or $k > i + 1$, and show that there exists another searcher on avenue $i$ or avenue $i + 1$ at time $t$. Assume, on the contrary, that there exists no searcher on avenue $i$ or avenue $i + 1$ at time $t$. Then, since

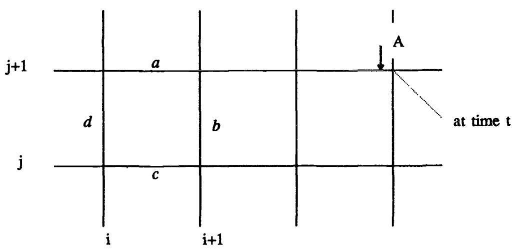


FIG. 6. *Illustration for Lemma* 3.

edges $b$ and $d$ cannot be seen by any searcher in $[t - \Delta, t]$, fugitive $Z$, which may be in edge $a$ at time $t - \Delta$, can move to either edge $b$ or edge $d$ in $[t - \Delta, t)$, as long as the maximum speed of $Z$ is not zero. Thus at least one of $b$ and $d$ is contaminated at time $t$, contradicting the assumption. □

THEOREM 3. *The time complexity of searching $G_N$ by a team of $O(1)$ searchers is $\Omega(N^2)$ if the maximum speed of the fugitive is not zero.*

*Proof.* If (1) the maximum speed of the fugitive is not zero, (2) the number of searchers is $O(1)$, and (3) the maximum speed of the searchers is one edge per unit time, then by Lemma 3 a constant number of blocks at most can become clear in unit time. Since (1) there are $(N-1)^2$ blocks in $G_N$, (2) initially all blocks except possibly one are contaminated, and (3) the search is complete if and only if either all blocks are clear simultaneously or one of the searchers sees the fugitive, we conclude that searching $G_N$ takes $\Omega(N^2)$ time units in the worst case. □

**3.3. Searching with limited communication.** By using algorithm SEARCH2 described below, $G_N$ can be searched by two searchers even if they can communicate only when they are on the same avenue or the same street. Initially, searchers $A$ and $B$ are in vertices $(1, 0)$ and $(0, 0)$, respectively. The traversal is illustrated in Fig. 7.

ALGORITHM SEARCH2.

```
Phase 1:  (c) $A$ and $B$ move north from $(1, 0)$ to $(1, N-1)$ and from $(0, 0)$ to
              $(0, N-1)$, respectively, in parallel.

Phase $k$:  $(k = 2, 4, 6, \cdots)$
          (a) $A$ stays in $(k-1, N-1)$, and $B$ moves east from $(k-2, N-1)$
              to $(k-1, N-1)$.
          (b) $B$ stays in $(k-1, N-1)$, and $A$ moves east from $(k-1, N-1)$
              to $(k, N-1)$.
          (c) $A$ and $B$ move south from $(k, N-1)$ to $(k, 0)$ and from
              $(k-1, N-1)$ to $(k-1, 0)$, respectively, in parallel.

Phase $k$:  $(k = 3, 5, 7, \cdots)$
          (a) $A$ stays in $(k-1, 0)$, and $B$ moves east from $(k-2, 0)$ to
              $(k-1, 0)$.
          (b) $B$ stays in $(k-1, 0)$, and $A$ moves east from $(k-1, 0)$ to $(k, 0)$.
          (c) $A$ and $B$ move north from $(k, 0)$ to $(k, N-1)$ and from $(k-1, 0)$
              to $(k-1, N-1)$, respectively, in parallel.
```

The time complexity of SEARCH2 is $N^2 + O(N)$. The correctness of SEARCH2 follows from an argument similar to the one in the proof of Theorem 1.

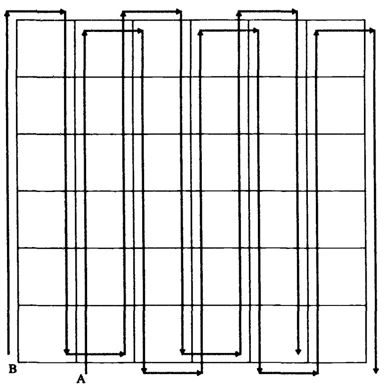


FIG. 7. *Traversal of $G_N$ determined by* SEARCH2.

**4. Capture problem.** In this section we present an $O(N^2)$ time algorithm for capturing a fugitive in $G_N$ by four searchers. It is assumed that the maximum speed of the searchers and the fugitive is one edge per unit time.

**4.1. A four-searcher algorithm.** Let $A$, $B$, $C$, and $D$ be the searchers, and $Z$ the fugitive. Fugitive $Z$ is said to be *captured* if one of the searchers occupies the same position as $Z$. It is assumed that the searchers can communicate with each other in real time irrespective of their positions. Furthermore, we assume that if at least one of the searchers can see $Z$, then the searchers can (1) determine the exact position of $Z$, and (2) move, without any delay, based on the observed movement of $Z$. For example, suppose that at time $t$ searchers $A$ and $B$ are in vertices (0, 0) and (1, 0), respectively, and $Z$ is on (1, 2) (see Fig. 8(a)). If $Z$ moves east to (2, 2) at a constant speed of one edge per unit time and reaches (2, 2) at time $t + 1$, then searcher $B$ can see the direction in which $Z$ moves at time $t$, and both $A$ and $B$ can start to move east at time $t$ and reach (1, 0) and (2, 0), respectively, at time $t + 1$ (see Fig. 8(b)). Of course, $Z$ may change directions in the middle of the edge between (1, 2) and (2, 2), and in this case $A$ and $B$ cannot know about it until $B$ finds out that $Z$ is not on (2, 2) at time $t + 1$. In the following, unless otherwise stated, the searchers always move at a speed of one edge per unit time.

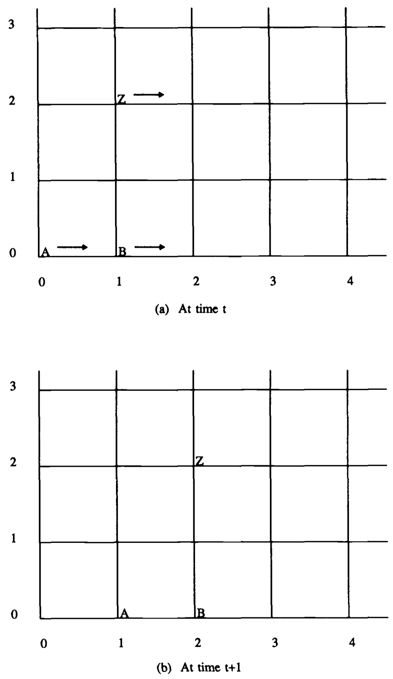


FIG. 8. *A and B can move at the moment Z moves.*

**4.1.1. Algorithm SEARCH3.** Since none of the searchers may be able to see $Z$ in the initial state, first they have to search for $Z$. Searching can be done by the following algorithm SEARCH3.

In SEARCH3, searchers $A$, $B$, $C$, and $D$ traverse $G_N$ in $N - 2$ phases, Phases 1 through $N-2$. Initially, $A, B, C$, and $D$ are in (0, 0), (1, 0), (2, 0), and (0, 0), respectively. The traversal is illustrated in Fig. 9.

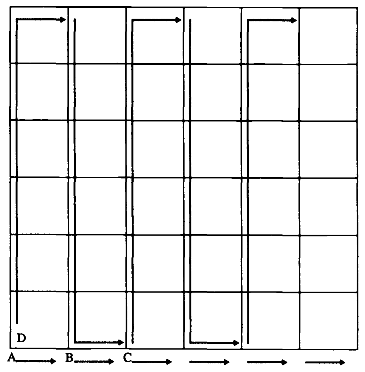


FIG. 9. *Traversal of $G_N$ determined by* SEARCH3.

ALGORITHM SEARCH3.

**Phase 1:** (b) $A$, $B$, and $C$ stay in $(0, 0)$, $(1, 0)$, and $(2, 0)$, respectively, and $D$ moves north from $(0, 0)$ to $(0, N-1)$, and then east to $(1, N-1)$.

**Phase $k$:** $(k = 2, 4, 6, \cdots)$

(a) $D$ stays in $(k-1, N-1)$, and $A$, $B$, and $C$ move east to $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, in parallel.

(b) $A$, $B$, and $C$ stay in $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, and $D$ moves south from $(k-1, N-1)$ to $(k-1, 0)$, and then east to $(k, 0)$.

**Phase $k$:** $(k = 3, 5, 7, \cdots)$

(a) $D$ stays in $(k-1, 0)$, and $A$, $B$, and $C$ move east to $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, in parallel.

(b) $A$, $B$, and $C$ stay in $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, and $D$ moves north from $(k-1, 0)$ to $(k-1, N-1)$, and then east to $(k, N-1)$.

The traversal is terminated at the moment the following condition LOCATE$(p, q)$ is satisfied for some integer $p = k$, $1 \leqq k \leqq N-2$, and $q = 0$.

*Condition* LOCATE$(p, q)$. The following conditions hold simultaneously:

(1) The positions of $A$, $B$, and $C$ are $(p-1, q)$, $(p, q)$, and $(p+1, q)$, respectively. ($p$ and $q$ are not necessarily integers.)

(2) The position $(x, y)$ of $Z$ satisfies $p-1 \leqq x \leqq p+1$ and $q \leqq y \leqq N-1$.

(3) The searchers know that $p-1 \leqq x \leqq p+1$ and $q \leqq y \leqq N-1$ hold.

(4) If $p$ is an integer, then the searchers know the exact value of $y$.

(5) If $x$ and $y$ are integers and either $x = p-1 > 0$ or $x = p+1 < N-1$, then $q$ is an integer.

If $A$, $B$, and $C$ are in positions $(p-1, q)$, $(p, q)$, and $(p+1, q)$, respectively, then the area consisting of all points $(x', y')$ such that $p-1 \leqq x' \leqq p+1$ and $q \leqq y' \leqq N-1$ is called the *critical area*, and is denoted by $CA(p, q)$ (see Fig. 10). Conditions (1) through (4) of LOCATE$(p, q)$ imply that the searchers know an approximate position of $Z$, which is in the critical area $CA(p, q)$. Condition (5) assures that whenever $Z$ is in a vertex and tries to move east or west in order to leave the current critical area, $A$, $B$, and $C$ are also in some vertices and can move east or west so that $Z$ will stay in new critical areas.

Since (1) $A$, $B$, and $C$ remain on street 0 and the critical area moves from west to east during the execution of SEARCH3; (2) the critical area is two edges wide east to west; (3) the searchers move at a speed of one edge per unit time; and (4) the maximum speed of $Z$ is one edge per time unit, $Z$ will eventually be in the critical area $CA(k, 0)$ for some integer $k$ when the searchers execute SEARCH3. (Without the fourth condition, $Z$ may only be detected, and LOCATE$(k, 0)$ may not hold for any integer $k$.) If $Z$ moves into $CA(k, 0)$ while $A$, $B$, and $C$ are stationary in $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, then searcher $C$ can see $Z$ at the moment and knows the exact position of $Z$ (see Fig. 11). If $Z$ is already in $CA(k, 0)$ when $A$, $B$, and $C$ move to $(k-1, 0)$, $(k, 0)$, and $(k+1, 0)$, respectively, then before $A$, $B$, and $C$ move east again, at least one of the searchers (including $D$) sees $Z$ and knows the exact position of $Z$. Of course, $Z$ may be captured during the execution of SEARCH3. Therefore we have the following lemma.

LEMMA 4. *If the searchers execute* SEARCH3, *eventually either $Z$ is captured or condition* LOCATE$(k, 0)$ *holds for some integer $k$, $1 \leqq k \leqq N-2$.* □

In the following section we denote by $T$ the time when LOCATE$(k, 0)$ holds for the first time for some integer $k$ during the execution of SEARCH3. Clearly $T = O(N^2)$.

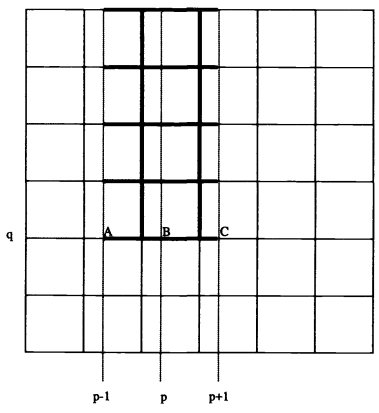


FIG. 10. *Critical area $CA(p, q)$.*

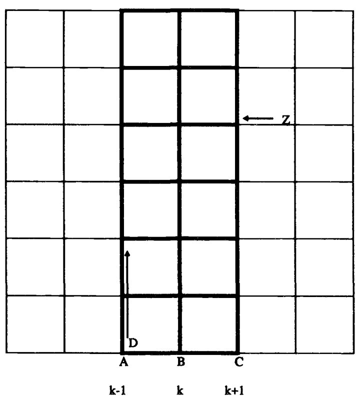


FIG. 11. *C sees Z.*

**4.1.2. Algorithm CHASE1.** At time $T$, $A$, $B$, and $C$ start executing algorithm CHASE1 and $D$ starts executing algorithm CHASE2 (CHASE2 is described in § 4.1.3).

ALGORITHM CHASE1.

Assume that LOCATE$(k, j)$ is true at time $t$ for some integers $k$ and $j$. Let $(x, y)$ be the position of $Z$ at $t$ (the exact value of $x$ may be unknown to the searchers). If (1) $k - 1 < x < k$ or $k < x < k + 1$ (i.e., $A$, $B$ and $C$ cannot see $Z$); or (2) either $x = k - 1 > 0$ or $x = k + 1 < N - 1$, and $y$ is not an integer, then $A$, $B$, and $C$ stay in their current positions until one of the four cases given below becomes applicable, at which moment they move as specified for each case until LOCATE$(k', j')$ becomes true again for some integers $k'$ and $j'$.

**Case 1:** If $x = k$, $x = 0$, or $x = N - 1$, then $A$, $B$, and $C$ move north and reach $(k - 1, j + 1)$, $(k, j + 1)$, and $(k + 1, j + 1)$, respectively, at time $t + 1$ (see Fig. 12(a), (b)).

**Case 2:** If $x = k - 1 > 0$, $y$ is an integer and $Z$ moves west at time $t$, then $A$, $B$, and $C$ move west and reach $(k - 2, j)$, $(k - 1, j)$, and $(k, j)$, respectively, at time $t + 1$ (see Fig. 13).

**Case 3:** If $x = k + 1 < N - 1$, $y$ is an integer and $Z$ moves east at time $t$, then $A$, $B$, and $C$ move east and reach $(k, j)$, $(k + 1, j)$, and $(k + 2, j)$, respectively, at time $t + 1$ (see Fig. 14).

**Case 4:** If $x = k - 1 > 0$ or $x = k + 1 < N - 1$, $y$ is an integer and $Z$ moves north or south, then $A$, $B$, and $C$ move north (and south if necessary) so that, at any moment $t' > t$, if $(x, y + h)$ is the position of $Z$ at $t'$ where $0 < |h| \leqq 1/2$, then $A$, $B$, and $C$ are in $(k - 1, j + |h|)$, $(k, j + |h|)$, and $(k + 1, j + |h|)$, respectively (see Fig. 15), and once $Z$ reaches position $(x, y \pm 1/2)$ and $A$, $B$, and $C$ reach $(k - 1, j + 1/2)$, $(k, j + 1/2)$, and $(k + 1, j + 1/2)$, respectively, $A$, $B$, and $C$ proceed to $(k - 1, j + 1)$, $(k, j + 1)$, and $(k + 1, j + 1)$, respectively, irrespective of the future movement of $Z$.

*Remark* 2. In Case 4 of CHASE1, $A$, $B$, and $C$ move north toward $Z$ as much as possible, as long as they can keep $Z$ in the critical areas. Since for example $Z$ may move west from a vertex on avenue $k - 1$ (if $A$ and $Z$ are both on avenue $k - 1$), keeping $Z$ in the critical areas requires that $A$, $B$, and $C$ are on some vertices whenever $Z$ reaches a vertex. This requirement is satisfied by the movement of $A$, $B$, and $C$ specified in Case 4. □

Lemma 5 implies that by using algorithm CHASE1, searchers $A$, $B$, and $C$ can keep $Z$ in critical areas at any moment, and furthermore, they may be able to get closer and closer to $Z$.

LEMMA 5. *If* LOCATE$(k, j)$ *is true at time* $t \geqq T$ *for some integers* $k$ *and* $j$, *then for any* $t' > t$, *either* $Z$ *is captured before* $t'$, *or* LOCATE$(p, q)$ *holds for some* $p$ *and* $q \geqq j$ *at* $t'$.

*Proof.* Assume that LOCATE$(k, j)$ holds for some integers $k$ and $j$ at time $t \geqq T$. Clearly, LOCATE$(k, j)$ continues to hold if none of the four cases of CHASE1 is applicable at $t$. Assume that one of the four cases is applicable at $t$. Let $t'$ be any time such that $t < t' \leqq t''$, where $t''$ is the earliest time after $t$ at which $A$, $B$, and $C$ reach some vertices. For Cases 1, 2, and 3, we have $t'' = t + 1$. Assume that $Z$ is not yet captured at $t'$. Let $(x, y)$ and $(x', y')$ be the positions of $Z$ at $t$ and $t'$, respectively.

*Case* 1. We consider the case in which $x = k$. Other cases are similar. At $t'$ the positions of $A$, $B$, and $C$ are $(k - 1, j + (t' - t))$, $(k, j + (t' - t))$, and $(k + 1, j + (t' - t))$, respectively (see Fig. 16). Since $k - 1 \leqq x' \leqq k + 1$ and $j + (t' - t) < y'$, conditions (1), (2), and (3) of LOCATE$(k, j + (t' - t))$ hold at $t'$. Since searcher $B$ stays on avenue $k$ between $t$ and $t'$, at $t'$ $B$ either sees $Z$ or knows the edge into which $Z$ has moved from avenue $k$. Thus condition (4) also holds. Finally, condition (5) is true at $t' < t + 1$ since $k - 1 < x' < k + 1$, and also at $t' = t + 1$ since $j + 1$ is an integer. Therefore LOCATE$(k, j + (t' - t))$ holds at $t'$, where $j + (t' - t) > j$.

*Case* 2. At $t'$ the positions of $A$, $B$, and $C$ are $(k - 1 - (t' - t), j)$, $(k - (t' - t), j)$, and $(k + 1 - (t' - t), j)$, respectively (see Fig. 17). Since $k - 1 - (t' - t) \leqq x' < k + 1 - (t' - t)$ and $j \leqq y - 1 < y'$, conditions (1), (2), and (3) of LOCATE$(k - (t' - t), j)$ hold at $t'$. Since (1) $k - (t' - t)$ is an integer only if $t' = t + 1$, and (2) if the searchers cannot see $Z$ at $t' = t + 1$ then the only possible value of $y'$ is $y$, condition (4) also holds. Finally, condition (5) is true at $t'$, since $j$ is an integer. Therefore LOCATE$(k - (t' - t), j)$ holds at $t'$.

*Case* 3. This case is similar to Case 2.

*Case* 4. Since $A$, $B$, and $C$ move north or south starting from their positions at $t$, conditions (1), (2), and (3) of LOCATE hold at $t'$. Since either $A$ or $C$ can see $Z$ at $t'$, condition (4) is also satisfied. Finally, condition (5) is satisfied at $t'$, since, as stated in Remark 2, Case 4 of CHASE1 assures that $A$, $B$, and $C$ are on some vertices whenever $Z$ reaches a vertex. Therefore LOCATE$(k, q)$ holds for some $q \geqq j$ at $t'$.

The lemma follows from above, since the same argument applies after LOCATE$(k', j')$ becomes true for some integers $k'$ and $j'$. □

Lemma 6 states that whenever $Z$ moves from one street to another, $A$, $B$, and $C$ can move north to the next higher numbered street and LOCATE$(k, j)$ becomes true for some integer $k$ and the next larger integer $j$.

LEMMA 6. *If for some $t$ and $t'$ such that $T \leqq t$ and $t + 1 \leqq t'$,*

(a) LOCATE$(p, q)$ *holds for some $p$ and $q$ with $q < N - 1$ at $t$,*

(b) *$Z$ is in vertex $(X, Y)$ at $t$,*

(c) *$Z$ is in vertex $(X, Y + 1)$ or $(X, Y - 1)$ at $t'$, and*

(d) *$Z$ is not in any vertex between $t$ and $t'$, then before $t' + 1$, either $Z$ is captured or* LOCATE$(p', \lfloor q \rfloor + 1)$ *holds for some integer $p'$.*

*Proof*. Assume that $Z$ is not captured before $t' + 1$. We consider the case in which $Z$ is in $(X, Y + 1)$ at $t'$. The other case is similar.

If $X = p$, $X = p - 1 = 0$, or $X = p + 1 = N - 1$, then $A$, $B$, and $C$ are executing Case 1 of CHASE1 at $t$, and reach $(p - 1, \lfloor q \rfloor + 1)$, $(p, \lfloor q \rfloor + 1)$, and $(p + 1, \lfloor q \rfloor + 1)$, respectively, by $t + 1 \leqq t'$. Thus by Lemma 5 LOCATE$(p, \lfloor q \rfloor + 1)$ holds before $t' + 1$.

Assume that $X = p - 1 > 0$. In this case $q$ is an integer by condition (5) of LOCATE$(p, q)$. By (b), (c), and (d) of this lemma, $A$, $B$, and $C$ start executing Case 4 of CHASE1 at $t$ and reach vertices $(p - 1, q + 1)$, $(p, q + 1)$, and $(p + 1, q + 1)$, respectively, by $t'$ at which $Z$ reaches $(X, Y + 1)$. Thus by Lemma 5 LOCATE$(p, q + 1)$ holds before $t' + 1$. The case in which $X = p + 1 < N - 1$ is similar.

Finally, assume that $p - 1 < X < p$ or $p < X < p + 1$. In this case $q$ is an integer and $A$, $B$, and $C$ are executing Case 2 or Case 3 of CHASE1. Assume Case 2 (Case 3 is similar). Let $t_1$ be the time when the execution of Case 2 is started, where $t - 1 < t_1 < t$. The only possible scenario is the following. At $t_1$, $Z$ moves west from vertex $(X, Y)$ and $A$, $B$, and $C$ start moving west from vertices $(X, q)$, $(X + 1, q)$, and $(X + 2, q)$, respectively. Then $Z$ returns to $(X, Y)$ at $t$, and moves north toward $(X, Y + 1)$. When the execution of Case 2 terminates at $t_1 + 1 < t'$, $Z$ is on avenue $X$ by (b), (c), and (d) of this lemma and $B$ is on $(X, q)$. So $A$, $B$, and $C$ start executing Case 1 of CHASE1 at $t_1 + 1$ and reach $(X - 1, q + 1)$, $(X, q + 1)$, and $(X + 1, q + 1)$, respectively, at $t_1 + 2$. Thus by Lemma 5 LOCATE$(X, q + 1)$ holds at $t_1 + 2 < t' + 1$. □

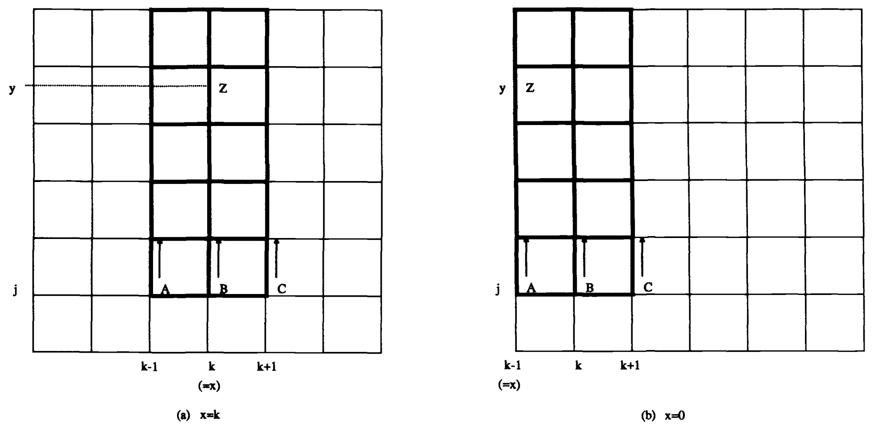


FIG. 12. *Case* 1 *of* CHASE1.

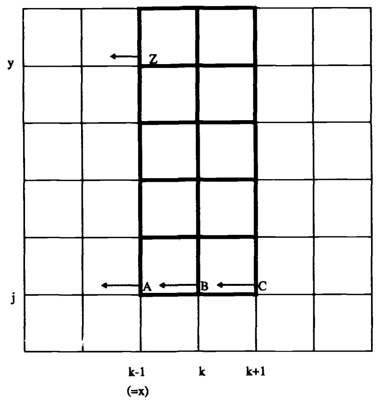


FIG. 13. *Case* 2 *of* CHASE1.

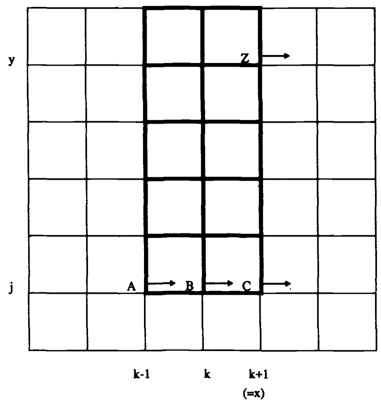


FIG. 14. *Case* 3 *of* CHASE1.

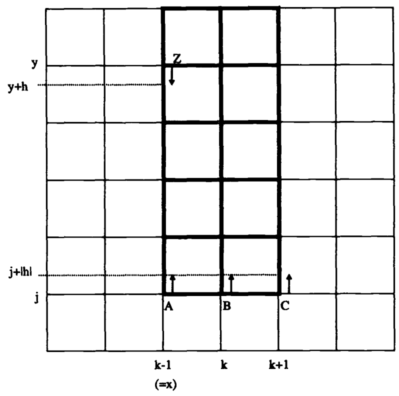


FIG. 15. *Case* 4 *of* CHASE1 *with* $0 < |h| \leqq 1/2$.

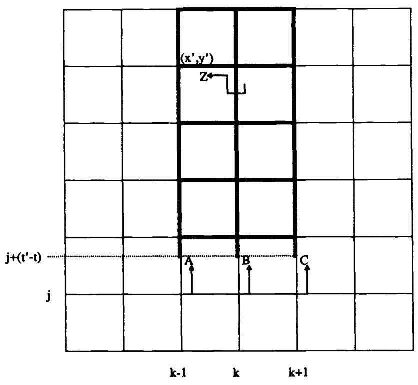


FIG. 16. *Case* 1 *of* CHASE1 *at* $t'$.

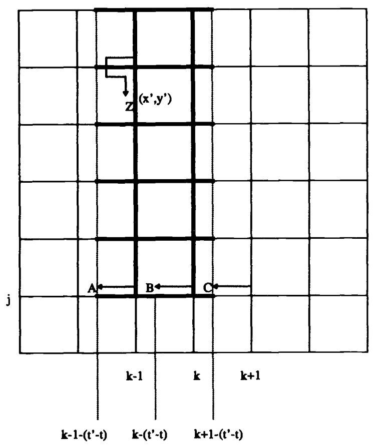


FIG. 17. *Case* 2 *of* CHASE1 *at* $t'$.

**4.1.3. Algorithm CHASE2.** Now we describe algorithm CHASE2 which searcher $D$ starts executing at time $T$. We say that $D$ *scans* street $j$ if $D$ visits all vertices on street $j$ either from west to east or from east to west (see Fig. 18).

ALGORITHM CHASE2.

The algorithm consists of two phases. It transits from Phase 1 to Phase 2 at the moment that $D$ reaches a position such that (a) the distance between $D$ and $Z$ is at most one, and (b) $D$ can see $Z$. (If (a) and (b) are already satisfied at $T$, then Phase 1 is not executed.)

**Phase 1:** $D$ repeatedly executes the following: $D$ stays in the current position as long as the value of $y$ is unknown where $(x, y)$ is the position of $Z$, and then scans streets $\lfloor y \rfloor$ and $\lceil y \rceil$ in any order. (If $y$ is an integer, then $D$ scans only street $y$.)

**Phase 2:** $D$ continuously moves toward $Z$. (This is possible, since the distance between $D$ and $Z$ is at most one and hence $D$ always knows the direction in which $Z$ moves.)

*Remark* 3. If $A$, $B$, and $C$ are moving west or east by Case 2 or Case 3 of CHASE1 and $D$ cannot see $Z$, then the searchers do not know the value of $y$, where $(x, y)$ is the position of $Z$. As is stated in the proof of Lemma 5, if Case 2 or Case 3 of CHASE1 is started at time $t$, then at time $t + 1$ either one of $A$, $B$, and $C$ can see $Z$ or the searchers know the edge in which $Z$ exists. Therefore in Phase 1 of CHASE2, $D$ does not continue to stay in its current position for one time unit or more. Thus $D$ completes a scan every $cN$ time units for some constant $c$. □

For each $j$ $(0 \leqq j \leqq N - 1)$, we denote by BACKBONE$(j)$ the area consisting of all points $(x, y)$ such that $j - 1 < y < j + 1$ (see Fig. 19). Lemma 7 states that if $Z$ stops moving from one street to another, then $Z$ is captured in $O(N)$ time units.

LEMMA 7. *If $Z$ stays in* BACKBONE$(j)$ *after time $t \geqq T$, then $Z$ is captured by time $t + c'N$ for some constant $c'$.*

*Proof.* Let $(x, y)$ be the position of $Z$ at $t$. Since $(x, y) \in$ BACKBONE$(j)$, we have either $\lfloor y \rfloor = j$ or $\lceil y \rceil = j$. Thus, by Remark 3, street $j$ will be scanned by $D$ by time $t + c''N$ for some constant $c''$. Then, since $Z$ stays in BACKBONE$(j)$, at some time $t' \leqq t + c''N$ the distance between $D$ and $Z$ becomes at most one and $D$ can see $Z$ (see Fig. 20). Then $D$ starts executing Phase 2 of CHASE2 at $t'$, and clearly $Z$ is captured by time $t + c'N$ for some constant $c'$. □

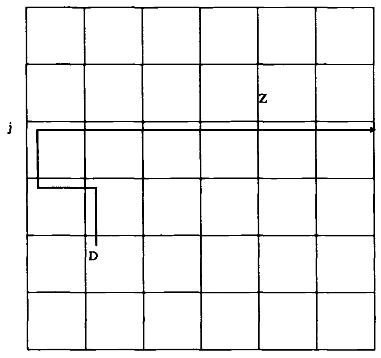


FIG. 18. *D scans street j.*

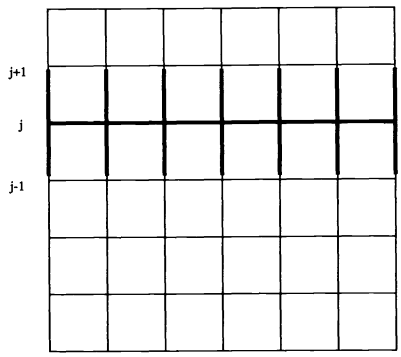


FIG. 19. BACKBONE($j$).

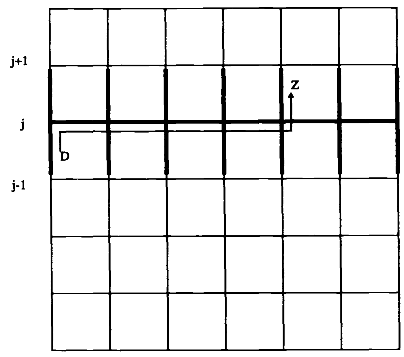


FIG. 20. *Illustration for Lemma* 7.

**4.2. Optimality.**

LEMMA 8. *There exists some time $t = T + O(N^2)$ such that either $Z$ is captured before $t$, or* LOCATE($K$, $N - 1$) *holds at $t$ for some integer $K$.*

*Proof.* By Lemma 7, $Z$ must move from one street to another at least every $kN$ time units for some constant $k$. Thus the lemma follows from Lemma 6. □

THEOREM 4. *The searchers can capture $Z$ in $O(N^2)$ time units by the method described above.*

*Proof.* If LOCATE($K$, $N - 1$) holds for some integer $K$, then the searchers can capture $Z$ in at most $1/2$ unit of time by moving toward $Z$. The theorem follows from this observation, $T = O(N^2)$, and Lemma 8. □

As was shown in Theorem 3, searching $G_N$ by four searchers takes $\Omega(N^2)$ time units in the worst case if the maximum speed of the fugitive is not zero. Therefore the method for capturing a fugitive by four searchers described above is asymptotically optimal with respect to the worst case time complexity for the case in which the fugitive is not immobile.

**5. Conclusions.** We have presented efficient algorithms for a pursuit-evasion problem in a grid $G_N$. It remains to be investigated (1) under what conditions two searchers are required to search $G_N$, and (2) whether four searchers are required to capture a fugitive in $G_N$ under the conditions assumed in this paper. Additional results on the pursuit-evasion problem under various conditions are presented in [16].

**Acknowledgment.** The authors wish to thank the anonymous referees for their helpful comments.

## REFERENCES

[1] M. AIGNER AND M. FROMME, *A game of cops and robbers*, Discrete Appl. Math., 8 (1984), pp. 1–12.

[2] T. ANDREAE, *Note on a pursuit game played on graphs*, Discrete Appl. Math., 9 (1984), pp. 111–115.

[3] ———, *On a pursuit game played on graphs for which a minor is excluded*, J. Combin. Theory Ser. B, 41 (1986), pp. 37–47.

[4] R. P. ANSTEE AND M. FARBER, *On bridged graphs and cop-win graphs*, J. Combin. Theory Ser. B, 44 (1988), pp. 22–28.

[5] P. FRANKL, *Cops and robbers in graphs with large girth and Cayley graphs*, Discrete Appl. Math., 17 (1987), pp. 301–305.

[6] J. E. HOPCROFT AND G. T. WILFONG, *Reducing multiple object motion planning to graph searching*, SIAM J. Comput., 15 (1986), pp. 768–785.

[7] C. E. KIM AND M. A. LANGTON, *Movement coordination for single-track robot systems*, J. Robotic Systems, 4 (1987), pp. 49–62.

[8] L. M. KIROUSIS AND C. H. PAPADIMITRIOU, *Searching and pebbling*, Theoret. Comput. Sci., 47 (1986), pp. 205–218.

[9] M. MAAMOUN AND H. MEYNIEL, *On a game of policemen and robber*, Discrete Appl. Math., 17 (1987), pp. 307–309.

[10] N. MEGIDDO, S. L. HAKIMI, M. R. GAREY, D. S. JOHNSON, AND C. H. PAPADIMITRIOU, *The complexity of searching a graph*, J. Assoc. Comput. Mach., 35 (1988), pp. 18–44.

[11] R. NOWAKOWSKI AND P. WINKLER, *Vertex-to-vertex pursuit in a graph*, Discrete Math., 43 (1983), pp. 235–239.

[12] T. D. PARSONS, *Pursuit-evasion in a graph*, Lecture Notes in Mathematics 642, Y. Alavi and D. R. Lick, eds., Springer-Verlag, Berlin, 1976, pp. 426–441.

[13] A. QUILLIOT, *A short note about pursuit games played on a graph with a given genus,* J. Combin. Theory Ser. B, 38 (1985), pp. 89–92.

[14] P. SPIRAKIS, *Moving many pebbles in a graph is polynomial time*, Tech. Rep. No. 93, Department of Computer Science, New York University, New York, 1983.

[15] K. SUGIHARA, I. SUZUKI, AND Y.-C. LEE, *Distributed algorithms for motion coordination of multiple robots in graphs*, Proc. 30th Midwest Symposium on Circuits and Systems, Syracuse, NY, 1987, pp. 652–655.

[16] K. SUGIHARA AND I. SUZUKI, *On a pursuit-evasion problem related to motion coordination of mobile robots*, Proc. 21st Hawaii International Conference on System Sciences, Kailua-Kona, Hawaii, 1988, pp. 218–226.

[17] C. K. YAP, *Coordinating the motion of several discs,* Tech. Rep. No. 105, Department of Computer Science, New York University, New York, 1984.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 145-155, May 1989

# UNIVERSAL GRAPHS FOR BOUNDED-DEGREE TREES AND PLANAR GRAPHS*

SANDEEP N. BHATT†, F. R. K. CHUNG‡, F. T. LEIGHTON§, AND ARNOLD L. ROSENBERG¶

**Abstract.** How small can a graph be that contains as subgraphs all trees on $n$ vertices with maximum degree $d$? In this paper, this question is answered by constructing such universal graphs that have $n$ vertices and bounded degree (depending only on $d$). Universal graphs with $n$ vertices and $O(n \log n)$ edges are also constructed that contain all bounded-degree planar graphs on $n$ vertices as subgraphs. In general, it is shown that the minimum universal graph containing all bounded-degree graphs on $n$ vertices with separators of size $n^\alpha$ has $O(n)$ edges if $\alpha < \frac{1}{2}$; $O(n \log n)$ edges if $\alpha = \frac{1}{2}$; $O(n^{2\alpha})$ edges if $\alpha > \frac{1}{2}$.

**Key words.** universal graphs, separators, trees, planar graphs

**AMS(MOS) subject classifications.** 05C, 65W, 68Q

## 1. Introduction.

Given a family $F$ of graphs, a graph $G$ is said to be $F$-universal if $G$ contains every graph in $F$ as a subgraph. A fundamental problem of interest is to determine how few edges a universal graph can have. Such problems are of interest in circuit design [V], data representation [CRS], [RSS], and parallel computing [BLe], [BCLR].

Let $f(F)$ denote the minimum number of edges in a graph that contains all graphs in $F$ as subgraphs. There is a large literature on universal graphs for various families of graphs. In the early 1960s, Rado first investigated universal graphs for infinite graphs [Ra]. Since then many results on this subject have been published. Here we give a list of some of the known results about universal graphs for various families of graphs.

(1) Moon [M] considered the universal graphs that contain all graphs on $n$ vertices as induced subgraphs. He established upper and lower bounds for the number of vertices in such universal graphs.

(2) Bondy [Bo] investigated universal graphs for the class $C_n$ of all cycles of length $\leqq n$; such universal graphs are called pancyclic and he showed that

$$n+\log_2(n-1)-1 < f(C_n) < n+\log_2(n-1)+\log^* n+O(1)$$

where $\log^* n$ denotes

$$\min\{k : \overbrace{\log\log\cdots\log}^{k} n < 2\}.$$

(3) Let $T_n$ denote the class of all trees on $n$ vertices. A lower bound of $\frac{1}{2}n \log n$ for $f(T_n)$ can be obtained by considering degree sequences of the universal graph. The upper

* Received by the editors July 1, 1988; accepted for publication (in revised form) October 24, 1988.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520. The research of this author was supported in part by National Science Foundation grant MIP-86-01885.

‡ Bellcore, Morristown, New Jersey 07960.

§ Department of Mathematics and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. The research of this author was supported in part by U.S. Air Force contract OSR-86-0076, Defense Advanced Research Projects Agency (DARPA) contract N00014-80-C-0622, U.S. Army contract DAAL-03-86-K-0171, and a National Science Foundation Presidential Young Investigator Award with matching funds from AT&T and IBM.

¶ Department of Computer Science, University of Massachusetts, Amherst, Massachusetts 01003. The research of this author was supported by National Science Foundation grants DCI-85-04308 and DCI-87-96236.

bound was improved by a series of papers [CG1], [CG2], [CG3], [CGP] and the best known upper bound is only a constant multiple of the lower bound [CG3]

$$\frac{1}{2}n \log n \leqq f(T_n) \leqq \frac{7}{\log 4} n \log n + O(n).$$

(4) One variation on universal graph problems is to require the universal graph to satisfy specified properties. In [CCG], it was shown that a minimum tree that contains all trees on $n$ vertices must have $n^{(1+o(1))\log n/\log 4}$ vertices, and that this is the best possible.

(5) A *caterpillar* is a tree with the property that its vertices of degree greater than one induce a path. Kimble and Schwenk [KS] first considered the problem of determining minimum caterpillars that contains all caterpillars on $n$ vertices, and they gave some estimates of the size of the universal caterpillar. In [CGS], it was shown that the minimum number of edges in such a universal caterpillar is within a constant factor of $n^2/\log n$.

(6) Let $E_n$ denote the class of all graphs with $n$ edges. It turns out that $E_n$-universal graphs contain many more edges than $T_n$-universal graphs. In fact, it was shown in [BCEGS] that

$$\frac{cn^2}{\log^2 n} < f(E_n) < (1+o(1))\frac{n^2 \log\log n}{\log n}.$$

In this paper, we consider universal graphs for the family $T_{n,d}$ of all trees on $n$ vertices with maximum degree $d$. We construct $T_{n,d}$-universal graphs on $n$ vertices with bounded degree (depending only on $d$). In related independent work, Friedman and Pippenger [FP] recently proved that an expander graph on $cn$ vertices with constant degree contains all trees on $n$ vertices with maximum degree $d$. (The constant $c$, which depends on $d$, is quite large.)

We will also consider universal graphs for the family $P_{n,d}$ of all planar graphs on $n$ vertices with maximum degree $d$. The $P_{n,d}$-universal graphs have $n$ vertices and $O(n \log n)$ edges, improving the previous bound of $O(n^{3/2})$ in [BCEGS].

In § 2, we use graph separators to construct universal graphs with $O(n)$ edges for the family of binary trees on $n$ vertices. Using similar techniques, we derive universal graphs for families of graphs of bounded degree with small separators. In particular, we obtain universal graphs with $O(n \log n)$ edges for bounded-degree planar graphs on $n$ vertices and universal graphs with $O(n)$ edges for bounded-degree outerplanar graphs on $n$ vertices. We also obtain $T_{n,d}$-universal graphs on $n$ vertices and $O(n)$ edges, but the maximum degree of these graphs is of order $O(\log n)$. To reduce the maximum degree, we modify our construction using expander graphs in § 3 and the resulting $T_{n,d}$-universal graphs have bounded degree. Section 4 concludes with further problems and remarks.

## 2. Universal graphs for families of graphs with small bisectors.

In a graph $G$ on $n$ vertices, a set $S$ of vertices is called a *bisector* if, by removing vertices in $S$ from $G$, the remaining graph can be partitioned into two exactly equal parts so that there is no edge joining a vertex from one part to the other.

Here we need a stronger notion of bisectors, called *k-bisectors*. When the vertices of a graph $G$ are colored in $k$ colors, a set $S$ of vertices is said to be a $k$-bisector if, by removing vertices in $S$ from $G$, the remaining graph can be partitioned into two exactly equal parts so that each part contains exactly equal numbers of vertices of each color, and there is no edge joining a vertex from one part to the other. Any tree on $n$ vertices has a bisector of size $c \log n$ and a $k$-bisector of size $ck \log n$ for some constant $c$ [C], [LT], [BL]. For binary trees, $c = (\log 3)^{-1} < 1$.

We first consider a simpler version of our problem, namely, for binary trees, that are trees with maximum degree three.

LEMMA 1. *The $n$ vertices of a binary tree $T$ can be mapped into a complete binary tree $C$ on no more than $2^q - 1$ vertices $(2^q - 1 \leqq n < 2^{q+1} - 1)$ so that at most* 6 $\log(n/2^{t-3})$ *vertices*[1] *of $T$ are mapped into a vertex of $C$ at distance $t$ from the root, and so that any two vertices adjacent in $T$ are mapped to vertices at most three apart in $C$.*

*Proof.* The idea is to recursively bisect $T$, placing the successive sets of bisector vertices within successively lower levels of $C$, until $T$ is decomposed into single vertices. For example, the vertices placed at the root of $C$ bisect $T$ into two subgraphs $T_1$ and $T_2$. Similarly, vertices mapped to the left child of the root bisect $T_1$, and vertices mapped onto the right child bisect $T_2$. In addition, at level $i$ of $C$ we map vertices of $T$ (that have not already been mapped within levels $i-1, i-2$) that are adjacent to vertices mapped at level $i-3$ of $C$. This ensures that vertices adjacent in $T$ will be mapped to vertices of $C$ distance three apart.

To keep the number of vertices of $T$ mapped to a level $i$ vertex in $C$ within the required bounds, we use 3-bisectors. The following procedure describes how this is done.

*Step* 0. Initialize every vertex of $T$ to color $A$, bisect $T$, and place the bisector vertices at the root (level 0) of $C$.

*Step* 1. For each subgraph created in the previous step, recolor every vertex adjacent to the bisector in the previous step with color 0, and place a 2-color bisector for the subgraph at the corresponding level-1 vertex of $C$.

*Step* 2. For each subgraph created in the previous step, recolor every vertex of color $A$ adjacent to the bisector in the previous step with color 1, and place a 3-color bisector for the subgraph at the corresponding level-2 vertex of $C$.

*Step* $t$ ($\log |T| \geqq t \geqq 3$). For each subgraph created in the previous step, place every vertex of color $t-1 \pmod 2$ at the corresponding level $t$ of $C$, recolor every vertex of color $A$ that is adjacent to a vertex mapped at the previous level with color $t-1 \pmod 2$ and place a 3-color bisector for the remaining subgraph at the corresponding level $t$ vertex of $C$.

To ensure the accuracy of Step $t$, it suffices to show $n_t \leqq 6\log(n/2^t) + 18$ for $3 \leqq t < \log|T|$. Since we have

$$\begin{aligned} n_t &\leqq 3\log\frac{n}{2^t} + \frac{1}{2}n_{t-3} \\ &\leqq 6\log\frac{n}{2^t} + 18, \end{aligned}$$

Lemma 1 is proved.

The analogous version for higher-degree trees and planar graphs can be proved in a very similar way, and the proofs are left to the reader. The main difference in proving these results is that vertices adjacent to previously mapped vertices are themselves only mapped at every $\theta(\log d)$ level instead of at every level. This way, only two or three colors are needed, and the bisector at every level stays small.

[1] Strictly speaking, we should use $\lfloor 6\log(n/2^{t-3})\rfloor$ instead of $6\log(n/2^{t-3})$. However, we will usually not bother with this type of detail since it has no significant effect on the arguments or results. Also, all logarithms henceforth are of base two.

LEMMA 2. *The vertices of a tree $T$ with maximum degree $d$ can be mapped into a complete binary tree $C$ on no more than $2^q - 1$ vertices $(2^q - 1 \leqq n < 2^{q+1} - 1)$ so that $O(\log(n/2^t))$ vertices of $T$ are mapped to a vertex of $C$ at distance $t$ from the root, and so that any two vertices adjacent in $T$ are mapped to vertices at most distance $O(\log d)$ $2 \log d + 2$ apart in $C$.*

LEMMA 3. *The vertices of a planar graph $G$ of maximum degree $d$ can be mapped into complete binary tree $C$ on $2^q - 1$ vertices $(2^q - 1 \leqq n < 2^{q+1} - 1)$ so that $O(\sqrt{n/2^t})$ vertices of $G$ are mapped to a vertex of $C$ at distance $t$ from the root, and so that any two vertices adjacent in $G$ are mapped to vertices at most distance $O(\log d)$ apart in $C$.*

A graph $G$ is said to have a *$k$-bisector function $f$* if any subgraph of $G$ on $m$ vertices has a $k$-bisector of size no more than $f(m)$. The preceding lemmas are all special cases of the following.

LEMMA 4. *Suppose $G$ on $n$ vertices with maximum degree $d$ has a $k$-bisector function $f$. The vertices of $G$ can be mapped into a complete binary tree $C$ on no more than $2^q - 1$ vertices where $2^q - 1 \leqq n < 2^{q+1} - 1$ so that $O(f(n/2^t))$ vertices of $G$ are mapped to a vertex of $C$ at distance $t$ from the root, and so that any two vertices adjacent in $G$ are mapped to vertices at most distance $O(k \log d)$ apart in $C$ if $k$ is large enough that*

$$2f(xd^{3k}) \leqq d^{3k-4} f(x) \quad \text{for all } x.$$

Although Lemma 4 looks somewhat complicated, it is a natural generalization of Lemmas 1–3, and we omit the proof. We can now construct universal graphs using the decomposition lemmas.

THEOREM 1. *The minimum universal graph for the family of all bounded degree trees on $n$ vertices has $n$ vertices and $O(n)$ edges.*

*Proof.* Using Lemma 2, we consider the graph with vertices grouped into clusters corresponding to the vertices in the complete binary tree $C$. A cluster corresponding to a vertex of level $t$ contains $O(\log(n/2^t))$ vertices. We connect all pairs of vertices in clusters with corresponding vertices within distance $O(\log d) = O(1)$ apart in $C$. By Lemma 2 the resulting graph is universal for the family of all trees with maximum degree $d$. The number $h(n)$ of edges in this graph is $O(n)$, since $h(n)$ satisfies the following recurrence inequality:

$$h(n) \leqq 2h\left(\frac{n}{2}\right) + c(\log n)^2$$

where $c$ is an appropriate constant depending on $d$.

The construction just described has $O(n)$ vertices. To obtain a universal graph with precisely $n$ vertices, we modify the embedding of Lemma 1 so that the same number of nodes of $T$ are wrapped to nodes in the same level of $C$. This is easy to do since we can always arbitrarily expand the bisector of any subtree to be within one of its maximum allowed value (which is the lesser of the number of nodes remaining and $O(\log(n/2^t))$ for nodes on level $t$ of $C$. The exact value of the maximum bisector is the same for all nodes on a level and depends on the parity of the number of nodes in the subgraphs at that level. Hence, the size of the bisectors at each level depends only on $n$, and the universal graph can be assumed to have precisely $n$ nodes.

THEOREM 2. *The minimum universal graph for the family of all bounded-degree planar graphs on $n$ vertices has $n$ vertices and $O(n \log n)$ edges.*

*Proof.* The construction is obtained by using Lemma 3 in similar fashion as in the proof of Theorem 1. The number of edges $h(n)$ satisfies

$$h(n) \leqq 2h\left(\frac{n}{2}\right) + cn$$

and, therefore, the minimum universal graph has $O(n \log n)$ edges.

THEOREM 3. *The minimum universal graph for a family of bounded-degree graphs on n vertices with bisector function* $f(x) = x^\alpha$ *has n vertices with* $O(n)$ *edges if* $\alpha < \frac{1}{2}$; $O(n \log n)$ *edges if* $\alpha = \frac{1}{2}$; $O(n^{2\alpha})$ *edges if* $\alpha > \frac{1}{2}$.

*Proof.* The construction follows from Lemma 4 together with the fact that the number $h(n)$ of edges satisfies

$$h(n) \leqq 2h\left(\frac{n}{2}\right) + c(f(n))^2.$$

THEOREM 4. *The minimum universal graph for the family of all bounded-degree outer-planar graphs on n vertices has n vertices and* $O(n)$ *edges.*

*Proof.* Since an outerplanar graph on $n$ vertices has a bisector of size $O(\log n)$, the result follows from Theorem 3.

**3. A bounded-degree universal graph for bounded-degree trees.** For the family of bounded-degree trees, the minimum universal graph has $n$ vertices and $O(n)$ edges as indicated in Theorem 2; however, the maximum degree is of order $\log n$. Although the number of the edges in this universal graph is within a constant factor of the optimum, its vertices have unbounded degree.

In this section we describe a construction for graphs on $n$ vertices with bounded-degree that are universal for all bounded degree trees on $n$ vertices. First we need a few definitions.

DEFINITION. A graph $G(V, E)$ is said to be *full* if for every $V' \subset V$, $|V'| \leqq |V|/2$, the number of edges between $V'$ and $V - V'$ is at least $|V'|$.

We observe that there is a constant $\delta$ such that for every $m$, there is a full graph on $m$ vertices with maximum degree $\delta$. Any expander graph can be used for constructing full graphs [AC], [LPS]. It was shown in [AC] that in any $\delta$-regular graph $G(V, E)$ with second largest eigenvalue $\lambda$, for every $V' \subset V$ with $|V'| = \alpha n$, the number $e(V')$ of edges contained in $V'$ satisfies

$$|e(V') - \tfrac{1}{2}\delta\alpha^2 n| \leqq \lambda\alpha(1-\alpha)n.$$

Therefore, there are at least $(\delta - 2\lambda)\alpha(1-\alpha)n$ edges between $V'$ and $V - V'$. As long as $(\delta - 2\lambda)/2 \geqq 1$, the graph $G$ is full. For large enough $\delta$, this is usually the case.

The universal graph $H$ on $n$ vertices is obtained as follows. For simplicity, we will assume that $n = 2^\alpha - 1$.

Start the construction with a complete binary tree on $n$ vertices. Then, add edges so that the vertices at level $k$ (a constant specified later) form a full graph on $2^k$ vertices. Repeat this for vertices at levels $2k$, $3k$, $\cdots$. Call the resulting graph $H_0$.

Next, add extra edges so that the vertices at levels $k$, $2k$, $\cdots$, $\log n - s$ ($k$ divides $\log n - s$ and $s$ is a constant specified later) collectively form a full graph. Call the resulting graph $H_1$.

Finally, insert an edge between any pair of vertices within distance $t$ of each other, where $t$ is a constant specified later. The resulting graph, denoted by $H$, is our universal graph. Observe that the maximum degree of any vertex in $H$ is no greater than

$(2\delta + 3)^t$ which, of course, is a constant because $\delta$ and $t$ are. We will show that $H$ is universal for the family of bounded-degree trees on $n$ vertices if $k$, $s$, and $t$ are properly chosen.

THEOREM 5. *For the family of trees on n vertices with maximum degree d, we can construct universal graphs on n vertices with bounded degree (that depends only on d).*

The proof of Theorem 5 is somewhat involved, and requires a few combinatorial facts concerning full graphs and trees. The intuition captured in the following lemmas may be understood as follows. Suppose that we have mapped a subset of the vertices of a tree $T$ within a graph $G$, and we next wish to map a vertex $v$ of $T$ onto a vertex of $G$ in such a way that it remains "close to" its neighbors that have already been embedded. If there is no place readily available, we can still find a suitable place for $v$ by "perturbing" the existing mapping slightly to make room for $v$. The "flow lemmas" establish conditions under which this can be done without dilating edges significantly.

LEMMA 5. *Let $G$ be a full graph with maximum degree $d$, and consider any assignment of packets to vertices of $G$ such that every vertex of $G$ is assigned at least $\lceil d/2 \rceil$ packets. Then, for any disjoint subsets $S$ and $T$ of vertices such that $|S| = |T|$, it is possible to redistribute the packets so that we have the following:*

(i) *Every packet either stays stationary or moves to a neighbor in $G$;*

(ii) *The number of packets in each vertex in $S$ decreases by one;*

(iii) *The number of packets in each vertex in $T$ increases by one; and*

(iv) *The number of packets in each vertex in $V - (T \cup S)$ remains the same.*

*Proof.* The lemma is proved with a simple max-flow/min-cut argument. Set up a flow problem with a supersource connected to each vertex in $S$ and a supersink connected to each vertex in $T$. Assign unit capacity to each edge. Because $G$ is full, there is a 0–1 flow with value $|S|$ between the source and sink. The flow determines a one-to-one correspondence (along with edge-disjoint paths) from the vertices in $S$ to the vertices in $T$. By moving one packet forward along each edge that has unit flow we can effect a reassignment of packets that satisfies conditions (ii)–(iv).

Since every vertex in the flow graph (with the supersource and supersink) has degree at most $d + 1$, at most $\lfloor (d+1)/2 \rfloor = \lceil d/2 \rceil$ packets will be removed from any vertex of $G$ during the reassignment process. Since every vertex of $G$ initially has $\lceil d/2 \rceil$ packets, no packet need ever move more than one step. Hence, the reassignment also satisfies the first condition.

LEMMA 6. *Let $G$ be a full graph on $n$ vertices with maximum degree $d$, and consider any assignment of packets to vertices of $G$ so that vertex $v_i$ has $a_i$ packets, where $a_i \geqq \lceil d/2 \rceil$ for $1 \leqq i \leqq m$. Then for any set of numbers $\{a'_i \mid 1 \leqq i \leqq m\}$ for which $a'_i \geqq \lceil d/2 \rceil$ for $1 \leqq i \leqq m$, it is possible to redistribute the packets so that we have the following:*

(i) *Every packet is reassigned to a vertex that is at distance at most* $\max_{1 \leqq i \leqq m} |a_i - a'_i|$ *from its original location in $G$; and*

(ii) *The number of packets assigned to $v_i$ changes from $a_i$ to $a'_i$, for all $1 \leqq i \leqq m$.*

*Proof.* Apply Lemma 5 for $\max_{1 \leqq i \leqq m} |a_i - a'_i|$ iterations, each iteration decreasing the maximum value of $|a_i - a'_i|$, $1 \leqq i \leqq m$, by one.

To establish Theorem 5 we use a decomposition strategy different from that in § 2. The following lemma is a simple extension of the $\frac{1}{3} : \frac{2}{3}$ separator theorem for binary trees and was observed previously in [BLe]. This can be generalized to arbitrary maximum degree $d$ via the $(1/(d+1), d/(d+1))$ separator theorem (see [C]).

LEMMA 7. *For every constant $p < \frac{1}{2}$, there exists a constant $q$ such that any $n$-vertex two-colored binary forest with $w$ vertices of color $A$ can be partitioned into two sets by the* *removal of $q$ edges so that each set has at least $\lceil pn \rceil$ vertices and at least $\lceil pw \rceil$ vertices of color $A$.*

We also require an additional, final lemma.

LEMMA 8. *Every binary tree $T$ on $n$ vertices can be embedded within $H_0$ so that we have the following*:

(i) *Vertices of $T$ are assigned to vertices in levels* $0, k, 2k, \cdots, \log n - s$ *of $H_0$*;

(ii) *Every vertex in levels* $0, k, 2k, \cdots, \log n - s$ *of $H_0$ is assigned at least* $\lceil d/2 \rceil$ *and at most $c_1$ vertices of $T$, where $c_1$ is some constant*;

(iii) *Vertices adjacent in $T$ are assigned to vertices in $H_0$ separated by distance at most $c_2$, for some constant $c_2$.*

Once Lemma 8 is established, it is easy to complete the proof of Theorem 5 as follows.

*Proof of Theorem* 5. First obtain the embedding of Lemma 8. Next, by Lemma 6 we can use the edges of $H_1 - H_0$ to reassign the vertices of $T$ within $H_1$ so that we have the following:

(i) Every vertex in levels $0, k, 2k, \cdots, \log n - s - k$ of $H_1$ is assigned $2^k - 1$ vertices of $T$;

(ii) Every vertex in level $\log n - s$ of $H_1$ is assigned $2^s - 1$ vertices of $T$; and

(iii) Vertices adjacent in $T$ are assigned to vertices in $H_1$ separated by distance at most $c_3$, where $c_3 \leqq c_2 + 2 \max(|2^s - 1 - \lceil d/2 \rceil|, |2^k - 1 - \lceil d/2 \rceil|)$.

At this point, we need only require that $s \geqq k$ and that $2^k - 1 \geqq \lceil d/2 \rceil$ so that the conditions of Lemma 6 are satisfied. Since $k$, $s$, $d$, $c_1$, and $c_2$ are all constants, we know that $c_3$ also is constant. We now reassign vertices one more time so that the mapping from $T$ to $H$ becomes one-to-one and onto. This is done by arbitrarily assigning the vertices of $T$ on levels $0, k, 2k, \cdots, \log n - s$ of $H_1$ to their immediate descendants. Once this is done, the maximum distance in $H_1$ between any two nodes adjacent in $T$ will be at most $c_3 + 2s$, which is constant. By setting $t = c_3 + 2s$ in the construction of $H$, this will mean that $T$ is a subgraph of $H$, thereby completing the proof of Theorem 5.

*Proof of Lemma* 8. We follow an approach similar to that in § 2. However, since we are allowed to place only $O(1)$ vertices of $T$ at any one vertex of $H_0$, we cannot afford to bisect the tree at each step because that may require placing $c \log n$ vertices of $T$ at the root of $H_0$ for some constant $c$. Therefore, instead of bisecting the tree at each step, we separate it into proportional size components using Lemma 7, and continually balance the sizes of components as the embedding proceeds towards lower levels of $H_0$.

Initially, color all the vertices of $T$ white. Then, pick any $\lceil d/2 \rceil$ vertices of $T$ and map them to the root (level 0) of $H_0$. Color *red* those vertices of $T$ that are adjacent to one or more of the vertices placed at the root of $H_0$. Next, fix $p$ with $\frac{1}{3} \leqq p < \frac{1}{2}$, and use Lemma 7 to partition the (as yet unmapped) vertices of $T$ into two sets, each with at least the fraction $p$ of the total number of unmapped vertices, and each with at least the fraction $p$ of the total number of red vertices (always rounded up to the nearest integer, of course). One of the sets is distributed to the left subtree of the root of $H_0$ and the other set to the right subtree. By Lemma 7, no more than $q$ edges connect vertices in the two sets.

No vertices of $T$ will be assigned to the next $k - 1$ levels of $H_0$, but we continue to partition $T$ into smaller and smaller sets. In particular, we first color vertices in the "left set" of $T$ (those unmapped vertices of $T$ assigned to the left subtree of $H_0$) that are adjacent to vertices in the right set. We then use Lemma 7 to partition the left and right sets each into two smaller subsets, one for each grandchild of the root. Continue in this fashion, coloring vertices red as they become adjacent to vertices in the opposite set and splitting the forests (sets) into smaller forests until we have distributed a forest to each vertex on the $k$th level of $H_0$.

Although the vertices are split into roughly equal proportions ($p : 1 - p$) at each level, the sizes of forests at the $k$th level could vary substantially (in fact, anywhere between $p^k$ and $(1 - p)^k$). Therefore, at this stage we balance the sizes of the forests assigned to each vertex by redistributing forests among vertices at level $k$. To achieve this balance, first use Lemma 7 to partition each forest into $\lceil d/2 \rceil$ subforests (but we do not distribute the subforests further down the tree). Next, we partition each subforest whose size is greater than $1/p$ times the size of the smallest subforest. Observe that this does not affect the size of the smallest subforest.

We are now ready to apply Lemma 6, with each subforest represented as a packet. In particular, we use Lemma 6 to redistribute subforests on the level so that every vertex ends up with an equal number of subforests (to within one). We then map all the red vertices of $T$ (i.e., those adjacent to vertices in different subforests) to the corresponding vertex of $H_0$ where the enclosing subforest is currently located, making sure to map at least $\lceil d/2 \rceil$ vertices of $T$ to each vertex on level $k$ in $H_0$. (If there are not enough red vertices, then we use some of the white vertices in the same subforest to make up the total. We show later that there are always enough vertices overall so that this is possible.)

After the mapping is completed for level $k$, we recolor red all white vertices of $T$ that are adjacent to vertices already mapped, and we henceforth regard the collection of subforests assembled at a single vertex of $H_0$ as a single forest. Next, we repeat the process used on levels $1, 2, \cdots, k$ for levels $k + 1, k + 2, \cdots, 2k, \cdots, \log n - s$, where $s$ is a constant yet to be specified. At every $k$th level, we rebalance and coalesce forests as on level $k$, and map all red vertices of $T$ to the corresponding vertices of $H_0$. At level $\log n - s$ all the unmapped vertices of $T$ (both red and white) are mapped directly to the corresponding vertices of $H_0$. Several details remain to be ironed out; however, it should be clear that vertices adjacent in $T$ are mapped to vertices which are at most $k$ levels apart in $H_0$.

The analysis needed to complete the proof is tedious, but not difficult. We start by letting $r_{ik}$ be the maximum number of red vertices in any forest after all partitioning, balancing, coalescing, mapping, and recoloring is done at level $ik$ of $H_0$. Similarly, let $z_{ik}$ be the number of vertices (both red and white) in the smallest forest at level $ik$.

We will prove by induction that, for $ik \leqq \log n - s$, $z_{ik} \geqq 2^{-ik}n/6$, and $r_{ik} \leqq r' = 96(1 + q)2^{-k}p^{-(k+\lceil \log \lceil d/2 \rceil \rceil + 1)}$.

Observing that $r' \geqq d\lceil d/2 \rceil$ for $k$ sufficiently large (in terms of $p$ and $d$), we note that both statements are trivially true for $i = 0$ and $n$ sufficiently large. We next calculate bounds for $r_{ik+k}$ and $z_{ik+k}$ to proceed with the inductive step.

By Lemma 7, we know that

$$r_{ik+1} \leqq (1-p)r_{ik} + 1 + q;$$

therefore, each forest at level $ik + k$ of $H_0$ has at most $(1 - p)^k r_{ik} + (1 + q)/p$ red vertices initially. The process of partitioning forests into subforests at level $ik + k$ cannot increase this value, but redistributing, coalescing, and recoloring certainly can. To measure their effect, we need to bound the number of subforests that are located at any vertex following redistribution. This of course depends on the overall number of subforests, which in turn depends on the size of the smallest subforest.

The size of the smallest subforest at level $ik$ is $z_{ik}$. Hence, the size of the smallest forest at level $ik + 1$ is at least $pz_{ik} - 1$. Applying the argument recursively, we find that the size of the smallest subforest at level $ik + k$ (after all the subdividing at this level is complete) is, for $p \leqq \frac{1}{2}$, at least

$$z_{ik}p^{k+\lceil \log \lceil d/2 \rceil \rceil} - (1-p)^{-1} \geqq p^{k+\lceil \log \lceil d/2 \rceil \rceil} 2^{-ik} n/6 - 2.$$

For sufficiently large $s$ (i.e., small enough $i$), this is at least $p^{k+\lceil \log \lceil d/2 \rceil \rceil} 2^{-ik} n/12$. Hence, the number of subforests at this level is no greater than $12 \times 2^{ik} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)}$. The maximum number of subforests located at any vertex after balancing is, therefore, no greater than

$$1 + 12 \times 2^{-k} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)} \leqq 24 \times 2^{-k} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)}.$$

Consequently, the maximum number of red vertices in any forest after rebalancing and coalescing is at most

$$((1-p)^k r_{ik} + (1+q)/p) 24 \times 2^{-k} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)}.$$

Since mapping and recoloring can increase this at most by a factor of two, we have

$$r_{ik+k} \leqq 48(1-p)^k r_{ik} 2^{-k} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)} + 48(1+q) 2^{-k} p^{-(k+\lceil \log \lceil d/2 \rceil \rceil)}.$$

By choosing $p > \frac{1}{3}$ so that $(1-p)/2p < 1$, we have that for $k$ sufficiently large (in terms of $p$ and $d$):

$$r_{ik+k} \leqq \tfrac{1}{2} r_{ik} + 48(1+q) 2^{-k} p^{-(k+1+\lceil \log \lceil d/2 \rceil \rceil)};$$

thus,

$$r_{ik+k} \leqq 96(1+q) 2^{-k} p^{-(k+1+\lceil \log \lceil d/2 \rceil \rceil)} = r',$$

as claimed.

We next complete the inductive step for $z_{ik+k}$. Since the largest and smallest subforests differ in size by at most a factor of $1/p$, the size of the smallest forest after balancing and coalescing is at least $p(n - r' 2^{ik+k}) 2^{-(ik+k)-1}$, the factor $\frac{1}{2}$ accounting for the fact that every vertex has the same number of packets to within one. After mapping and recoloring, the size of the smallest forest is

$$z_{ik+k} \geqq \frac{p}{2}(n - r' 2^{ik+k}) 2^{-(ik+k)} - r'.$$

With some additional calculations it can be checked that this is at least $2^{-(ik+k)} n/6$ for $p > \frac{1}{3}$ and $s$ sufficiently large, thereby completing the proof of the claim.

By choosing $s$ sufficiently large, we have shown that every vertex at levels $0, k, \cdots, \log n - s - k$ of $H_0$ is assigned at least $\lceil d/2 \rceil$ and at most $r'$ vertices of $T$. Since $s$ is constant, every vertex at level $\log n - s$ of $H_0$ is assigned between $\lceil d/2 \rceil$ and $c_1$ vertices, where $c_1$ is some constant bigger than $r'$. Moreover, vertices of $T$ are assigned only to vertices in levels $0, k, \cdots, \log n - s$ of $H_0$. Hence, it remains only to show that vertices adjacent in $T$ are assigned to vertices in $H_0$ that are separated by distance at most $c_2$, for some constant $c_2$. We already know that $c_2$ is at most $k$ plus the distance subforests are allowed to move during the rebalancing step at every $k$th level. By Lemma 6, this distance is at most the largest number of subforests at any vertex before rebalancing. By the construction, this is at most some constant determined by $p$, $d$, $k$, and $s$. This completes the proof of Lemma 8 and Theorem 5.

## 4. Further problems and remarks.

While the universal graph for bounded-degree trees is optimal (within a constant factor), the universal graph for bounded-degree planar graphs on $n$ vertices has $O(n \log n)$ edges. On the other hand, the best known lower bound for the number of edges is still $cn$. It is of interest to close up the gap.

One variation of the universal graph problem is to require the universal graph to be of some specified type. For example, in [CCG] universal trees that contain all trees with at most $n$ nodes were considered. Relatively little is known about universal planar graphs that contain all planar graphs on $n$ vertices.

This work is heavily motivated by simulation of graph families in various host networks with small *dilation* (i.e., adjacent vertices are mapped into nearby vertices) and small *expansion* (i.e., the ratio of the size of host graph and the maximum size of graphs in the family is small). The decomposition lemma (Lemma 1 in § 2) for binary trees also provides optimal embeddings of binary trees within other structures. For example, we can show that every $n$-vertex binary tree can be embedded within an $n$-vertex complete binary tree with expansion 1 and dilation $O(\log \log n)$. This settles a conjecture of Hong, Mehlhorn, and Rosenberg [HMR] who showed a lower bound of $\Omega(\log \log n)$ for this problem. By embedding a complete binary tree within the shuffle exchange graph with expansion 1 and dilation 2, we obtain $O(\log \log n)$ dilation for arbitrary trees embedded within the shuffle-exchange graphs. Similarly, we have recently shown that an $n$-vertex binary tree can be embedded with constant expansion and dilation within the butterfly network [BCHLR]. Finally, we have shown that all binary trees can be embedded in a hypercube with expansion 1 and dilation 10 [BCLR].

## REFERENCES

[AC] N. Alon and F. R. K. Chung, *Explicit construction of linear sized tolerant networks*, Discrete Mathematics, 72 (1988), pp. 15–20.

[BCEGS] L. Babai, F. R. K. Chung, P. Erdös, R. L. Graham, and J. Spencer, *On graphs which contain all sparse graphs*, Ann. Discrete Math., 12 (1982), pp. 21–26.

[BL] S. N. Bhatt and F. T. Leighton, *A framework for solving* VLSI *graph layout problems*, J. Comput. Systems Sci., 28 (1984), pp. 300–343.

[BLe] S. N. Bhatt and C. E. Leiserson, *How to assemble tree machines*, in Advances in Computing Research 2, F. Preparata, ed., JAI Press, New York, 1984.

[BCLR] S. N. Bhatt, F. R. K. Chung, T. Leighton, and A. L. Rosenberg, *Optimal simulations of tree machines*, in Proc. 27th Annual IEEE Symposium on Foundations of Computer Science, Toronto, 1986, pp. 274–282.

[BCHLR] S. N. Bhatt, F. R. K. Chung, J.-W. Hong, T. Leighton, and A. L. Rosenberg, *Optimal simulations by butterfly networks*, in Proc. 20th Annual ACM Symposium on Theory of Computing, Chicago, 1988, pp. 192–204.

[Bo] J. A. Bondy, *Pancyclic graphs*, J. Combin. Theory Ser. B, 11 (1971), pp. 80–84.

[C] F. R. K. Chung, *Separator theorems and their applications*, to appear.

[CCG] F. R. K. Chung, D. Coppersmith, and R. L. Graham, *On trees which contain all small trees*, in The Theory of Applications of Graphs, G. Chartrand, ed., John Wiley, New York, 1981, pp. 265–272.

[CG1] F. R. K. Chung and R. L. Graham, *On graphs which contain all small trees*, J. Combin. Theory Ser. B, 24 (1978), pp. 14–23.

[CG2] ———, *On universal graphs*, Ann. New York Acad. Sci., 319 (1979), pp. 136–140.

[CG3] ———, *On universal graphs for spanning trees*, Proc. London Math. Soc., 27 (1983), pp. 203–211.

[CGP] F. R. K. Chung, R. L. Graham, and N. Pippenger, *On graphs which contain all small trees* II, in Proc. 1976 Hungarian Colloquium on Combinatorics, North Holland, Amsterdam, 1978, pp. 213–223.

[CGS] F. R. K. Chung, R. L. Graham, and J. Shearer, *Universal caterpillars*, J. Combin. Theory Ser. B, 31 (1981), pp. 348–355.

[CRS] F. R. K. Chung, A. L. Rosenberg, and L. Snyder, *Perfect storage representations for families of data structures*, SIAM J. Algebraic Discrete Math., 4 (1983), pp. 548–565.

[FP] J. Friedman and N. Pippenger, *Expanding graphs contain all small trees*, Combinatorica, 7 (1987), pp. 71–76.

[HMR] J.-W. Hong, K. Mehlhorn, and A. L. Rosenberg, *Cost trade-offs in graph embeddings with applications*, Assoc. Comput. Mach., 30 (1983), pp. 709–728.

[KS] R. J. Kimble and A. J. Schwenk, *On universal caterpillars*, in The Theory and Applications of Graphs, Gary Chartrand, Yousef Alavi, Donald L. Goldsmith, Linda Lesniak, and Don R. Lick, eds., John Wiley, New York, 1981, pp. 437–447.

[LPS] A. Lubotzky, R. Phillips, and P. Sarnak, *Ramanujan graphs*, Combinatorica, 8 (1988) pp. 261–278.

[LT] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[M] J. W. Moon, *On minimal n-universal graphs*, Proc. Glasgow Math. Soc., 7 (1965), pp. 32–33.

[Ra] R. Rado, *Universal graphs and universal functions*, Acta Arith., 9 (1964), pp. 331–340.

[Ro] A. L. Rosenberg, *Issues in the study of graph embeddings*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, New York, 1981, pp. 150–176.

[RSS] A. L. Rosenberg, L. J. Stockmeyer, and L. Snyder, *Uniform data encodings*, Theoret. Comput. Sci., 2 (1980), pp. 145–165.

[V] L. G. Valiant, *Universality considerations in* VLSI *circuits*, IEEE Trans. Comput., C-30 (1981), pp. 135–140.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 156-175, May 1989

# VERTEX TYPES IN BOOK-EMBEDDINGS*

JONATHAN F. BUSS†, ARNOLD L. ROSENBERG‡, AND JUDSON D. KNOTT§

**Abstract.** A new measure of the complexity of a book-embedding of a simple undirected graph, the number of *vertex types* in the embedding, is studied. The *type* of a vertex $v$ in a $p$-page book-embedding is the $p \times 2$ matrix of nonnegative integers

$$\tau(v) = \begin{pmatrix} L_1 & R_1 \\ L_2 & R_2 \\ & \vdots \\ L_p & R_p \end{pmatrix},$$

where $L_i$ (respectively, $R_i$) is the number of edges incident to $v$ that connect on page $i$ to vertices lying to the left (respectively, to the right) of $v$. The number of types in a book-embedding relates to the amount of logic necessary to realize fault-tolerant arrays of processors using one specific design methodology. Three sorts of issues regarding vertex types in book-embeddings are studied. A number of techniques for bounding the type-numbers of a variety of graphs are developed; the relationships between typenumber and other graph properties, such as book thickness, are investigated; and the problem of minimizing the typenumber of a graph is considered. Finding that problem to be NP-complete, the problem of finding a (cyclic) rotation of a given book-embedding that minimizes the number of vertex types is studied.

**Key words.** fault-tolerance, VLSI design, graph embedding, outerplanar graphs, trees, NP-completeness

**AMS(MOS) subject classifications.** primary 68R10; secondary 05C10, 68Q35

## 1. Introduction.

A *book* is a set of half-planes (the *pages* of the book) that share a common boundary line (the *spine* of the book). An *embedding* of a simple undirected graph[1] $G$ in a book consists of an ordering of the vertices of $G$ along the spine of the book, together with an assignment of each edge of $G$ to a page of the book, in which edges assigned to the same page do not cross. We shall always assume that a graph is simple and undirected. A (*cyclic*) *rotation* of a given book-embedding is obtained by cyclically rotating the vertices along the spine, while retaining all assignments of edges to pages.

There are three germane measures of the quality of a book-embedding:

(1) The *thickness* (= number of pages) of the book;

(2) The individual and cumulative *widths of the pages* (= the cutwidths of the edges on the various pages, and the cutwidth of the entire embedding);

(3) The *number of distinct vertex types*: Given a $p$-page book-embedding of a graph $G$, each vertex $v$ of $G$ has an associated $p \times 2$ matrix of nonnegative integers, called its *type*,

$$\tau(v) = \begin{pmatrix} L_1 & R_1 \\ L_2 & R_2 \\ & \vdots \\ L_p & R_p \end{pmatrix}. \tag{1}$$

* Received by the editors March 17, 1987; accepted for publication (in revised form) September 7, 1988. This research was supported in part by Semiconductor Research Corporation contract 85-02-054.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

‡ Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003. The work of this author was supported in part by National Science Foundation grant DMC-85-04308.

§ Department of Computer Science, Duke University, Durham, North Carolina 27706.

[1] An undirected graph is *simple* if a pair of vertices is connected by at most one edge.

$L_i$ (respectively, $R_i$) is the number of edges incident to $v$ that connect on page $i$ to vertices lying to the left (respectively, to the right) of $v$.

Note that any cyclic permutation (or, *rotation*) of a $p$-page book-embedding is another $p$-page book-embedding.

Henceforth we refer to the *book-thickness* (also called the *pagenumber*), the *pagewidth*, and the *typenumber* of a book-embedding and, via minimization over all book-embeddings, of a graph $G$.

Most work on book-embeddings has focused solely on book-thickness [1], [2], [5], [8], [10], [15]; some have included pagewidth on their list of concerns [3], [4], [9], [13]. The present paper concentrates on the third cost measure. We study techniques for bounding the typenumbers of graphs by studying a variety of specific graphs (§ 3); we then investigate the relationships between typenumber and a number of other structural properties of graphs (§ 4); we finally consider the problem of minimizing the typenumber of a graph. We show that the problem of finding a typenumber-minimal one-page embedding of an outerplanar graph is NP-complete. We then fall back to the problem of minimizing the typenumber of a given book-embedding, by means of a (cyclic) rotation. We find a rather efficient algorithm for this minimization problem. We also demonstrate the need for such an algorithm by considering the case of complete $d$-ary trees: the typenumber of any such tree is three; however, a randomly chosen rotation of the preorder book-embedding of the tree has, with probability approaching one, $d + 4$ vertex types.

Motivating our study is the DIOGENES methodology for designing fault-tolerant VLSI processor arrays [12], [3]. The methodology views the desired array as an undirected graph, with vertices representing processing elements and edges representing communication links; the design process operates in two stages: First, the graph representing the desired array is embedded in a book; then, the book-embedding is converted to an efficient fault-tolerant layout of the array. The significance of the notion of vertex type is that the type of a vertex "tells it" what role to play in the fault-free processor array. Thus, the base-two logarithm of the number of vertex types is the number of control bits per processing element needed to configure the array to its fault-free format.

Before turning to the main results of the paper, we present some easily verified basic facts about vertex types and book-embeddings that are useful in the sequel.

PROPOSITION 1 [1]. *The graph G admits a 1-page book-embedding if and only if G is outerplanar.*

The next result indicates the fundamental nature of vertex types.

PROPOSITION 2. *A book-embedding is determined uniquely by the sequence of vertex types it induces.*

*Sketch of proof.* Since edges on the same page of a book-embedding cannot cross, we can recreate the entire embedding by reading off (from the sequence of vertex types) how many edges leave each vertex to the right and to the left on each page of the book and matching these "dangling edges" up in a left-to-right scan of the sequence. For instance, if the sequence of vertices in the book-embedding is

$$v_1, v_2, \cdots, v_n$$

and if on Page $k$ we have the following:

- Vertex $v_i$ has $e \geqq 1$ edges leaving to the right;
- Vertex $v_j$, $j > i$, has at least one edge leaving to the left;
- At most $e - 1$ vertices $v_l$, $i < l < j$, have edges leaving to the left;
- No vertex $v_m$, $i < m < j$, has an edge leaving to the right;

then we know that vertices $v_i$ and $v_j$ are connected by an edge on page $k$. □

The next two results present general upper and lower bounds on the typenumber of an arbitrary graph and of an outerplanar graph.

A nonzero vertex type of the form (1) is a *source* if all $L_i = 0$ and is a *sink* if all $R_i = 0$. A source or a sink is a *one-sided* vertex type; a vertex type that is neither a source nor a sink is *two-sided.*

PROPOSITION 3. *Every graph $G$ having a connected component with at least two vertices has typenumber $t \geqq 2$.*

*Sketch of proof.* Under the stated hypotheses, every book-embedding of $G$ must have a source and a sink. □

We denote by $n_G$ the number of vertices of the graph $G$ and by $n_G(d)$ the number of degree-$d$ vertices of $G$.

PROPOSITION 4. (a) *In any p-page book-embedding of a graph $G$, the degree-d vertices of $G$ can assume no more than*

$$\min\left(n_G(d), \binom{2p+d-1}{d}\right) \tag{2}$$

*vertex types. Hence, if $D_G$ is the set of all distinct vertex-degrees of the graph $G$, then the typenumber of the given book-embedding can be no more than*

$$\sum_{k \in D_G} \min\left(n_G(k), \binom{2p+k-1}{k}\right). \tag{3}$$

(b) *The typenumber of any book-embedding of $G$ must be at least $|D_G|$.*

(c) *If $G$ is a biconnected outerplanar graph, then the typenumber of any 1-page book-embedding of $G$ cannot exceed*

$$2 + \sum_{k \in D_G} \min(n_G(k), k-1).$$

*Sketch of proof.* (a) The two possibilities in (2) hold, respectively, since each vertex has exactly one type, and since each vertex type can be viewed as a partition of the integer $d$ into $2p$ nonnegative parts. Equation (3) is a direct consequence of (2).

(b) The lower bound is immediate since for any degree-$d$ vertex $v$, $\tau(v)$, as specified in (1), must satisfy $\sum_{i=1}^{p} L_i + \sum_{i=1}^{p} R_i = d$.

(c) A direct instantiation of (3) would yield the bound

$$\sum_{k \in D} \min(n_G(k), k+1).$$

We can reduce this total by noting that a 1-page book-embedding of a biconnected graph must have *exactly* one source and one sink, as we now verify. □

LEMMA 1. *A 1-page book-embedding of a biconnected outerplanar graph has precisely one source and one sink.*

*Sketch of proof.* Assume that the embedding had more than one source (sinks yielding to a symmetric argument). The rightmost neighbors of all but the leftmost source are easily seen to be cut vertices, contradicting the biconnectivity of $G$. □

**2. The typenumbers of specific graphs.** We derive upper and lower bounds on the typenumbers of (book-embeddings of) a variety of families of graphs.

**Stars.** The $d$-ary *star* is the graph with $d + 1$ vertices and with edges connecting one of these vertices (the *root*) to all the others (the *leaves*).

PROPOSITION 5. *A connected graph G with at least two vertices has typenumber two if and only if G is a star.*

*Sketch of proof.* The one-page book-embedding of a star that places the root to one side (say the left) of all the leaves has two vertex types: the root has type $(0, d)$, and every leaf has type $(1, 0)$.

Conversely, let $G$ be a connected graph with typenumber two. By Proposition 3, $G$ has at least one source and at least one sink. Since $G$ has typenumber two, it has precisely one source, say

$$\begin{pmatrix} 0 & R_1 \\ 0 & R_2 \\ & \vdots \\ 0 & R_p \end{pmatrix}$$

and precisely one sink, say

$$\begin{pmatrix} L_1 & 0 \\ L_2 & 0 \\ & \vdots \\ L_p & 0 \end{pmatrix}.$$

If both $\sum_{i=1}^{p} L_i > 1$, and $\sum_{i=1}^{p} R_i > 1$, then $G$ would not be a simple graph, as we can verify easily by considering the leftmost sink in the book-embedding and the source immediately to its left. If $G$ is connected, and at least one of $\sum_{i=1}^{p} L_i$ and $\sum_{i=1}^{p} R_i$ equals one, then $G$ is a star. □

**Complete trees.** For integers $d$, $h > 0$, the *height-h complete d-ary tree* $T_{d,h}$ is the rooted tree in which every nonleaf node has $d$ sons, and all root-to-leaf paths have length $h$ (measured in number of vertices). The *preorder book-embedding* of $T_{d,h}$ is the 1-page book-embedding whose vertex-ordering is given recursively by

**0.** the root of $T_{d,h}$, to the left of
**1.** the vertices of the leftmost copy of $T_{d,h-1}$ in preorder, to the left of
**2.** the vertices of the second leftmost copy of $T_{d,h-1}$ in preorder, to the left of
. . .
$d$ the vertices of the rightmost copy of $T_{d,h-1}$ in preorder.

Note that $T_{d,2}$ is the $d$-ary star.

PROPOSITION 6. (a) *A 1-page book-embedding of a complete d-ary tree can have typenumber at most*

$$\begin{array}{ll} 1 & \text{if } h=1, \\ 3 & \text{if } h=2, \\ d+3 & \text{if } h=3, \\ d+5 & \text{if } h\geqq 4. \end{array}$$

(b) *For any arity $d \geqq 2$, the preorder book-embedding of $T_{d,h}$ (which is a 1-page book-embedding) has typenumber one when $h = 1$, typenumber two when $h = 2$, and typenumber three when $h \geqq 3$. No book-embedding of $T_{d,h}$ has smaller typenumber.*

*Proof.* Item (a) is immediate from Proposition 4, since the root of $T_{d,h}$ is the unique node of degree $d$, all $(d^{h-1} - d)/(d-1)$ interior nodes have degree $d + 1$, and all $d^h - 1$ leaf nodes have degree one.

(b) In the preorder book-embedding: the root has type $(0, d)$; each internal node has type $(1, d)$; and each leaf node has type $(1, 0)$. The typenumber-minimality of the preorder book-embedding follows from the lower bound in Proposition 4. □

**Complete graphs.** The *$n$-vertex complete graph $K_n$* has $n$ vertices, every two of which are connected by an edge.

PROPOSITION 7. *Typenumber $(K_n) = n$.*

*Sketch of Proof.* In any book-embedding of $K_n$, the $i$th vertex from the left is unique in having precisely $i - 1$ edges going to the left; hence, all vertices have distinct types. □

**Complete bipartite graphs.** For integers $m, n \geqq 1$, the *complete bipartite graph $K_{m,n}$* has $m$ *input* vertices and $n$ *output* vertices, and edges connecting each input with each output.

PROPOSITION 8. *Typenumber $(K_{m,n}) = 1 + \min(m, n)$. The book-embedding achieving this typenumber is unique, up to rotation.*

*Proof.* We show first that Typenumber $(K_{m,n}) \leqq 1 + \min(m, n)$. Say, with no loss of generality, that $m \leqq n$. Consider the $m$-page book-embedding of $K_{m,n}$ that places all inputs to the left of all outputs and that uses page $i$ $(1 \leqq i \leqq m)$ for the star that connects the $i$th input vertex to all of the output vertices. In this embedding, each input vertex has a distinct type, and all output vertices have the same type, for a total of $1 + \min(m, n)$ types.

We complete the proof by showing that Typenumber $(K_{m,n}) \geqq 1 + \min(m, n)$, and that the book-embedding achieving this bound (which is the one described in the preceding paragraph) is unique, up to rotation.

Note first that the bound and the uniqueness are trivial when $\min(m, n) = 1$, for then $K_{m,n}$ is a star. We, therefore, assume henceforth that $\min(m, n) > 1$.

Focus on a fixed book-embedding of $K_{m,n}$. Say that there are two input vertices, $u$ and $v$, that have the same type in the embedding. (Our conclusions will translate by symmetry to output vertices.)

CLAIM 1. *No output lies between $u$ and $v$ in the book-embedding.*

If $u$ and $v$ were separated by an output, they would have different numbers of leftgoing edges, and hence different types.

CLAIM 2. *For each output $w$, the edges from $w$ to both $u$ and $v$ lie on the same page.*

CLAIM 3. *For each pair of outputs $w$ and $x$, the edges from $u$ to $w$ and $x$ lie on different pages; the same is true of the edges from $v$ to $w$ and $x$.*

We verify Claims 2 and 3 simultaneously. By Claim 1, no output lies between inputs $u$ and $v$ in the book-embedding. Focus, therefore, on the $k$ outputs that lie to the left of both $u$ and $v$ and on the $l$ outputs that lie to the right of $u$ and $v$. Say that the $k$ left-hand outputs are, from left to right,

$$o_k, o_{k-1}, \cdots, o_1$$

and that the $l$ right-hand outputs are, from left to right,

$$\hat{o}_1, \hat{o}_2, \cdots, \hat{o}_l.$$

Say that, for each $i \in \{1, 2, \cdots, k\}$, input $u$'s edge to output $o_i$ resides on page $p_i$, and input $v$'s edge to output $o_i$ resides on page $q_i$; say, moreover, that, for each $j \in \{1, 2, \cdots, l\}$, input $u$'s edge to output $\hat{o}_j$ resides on page $\hat{p}_j$, and input $v$'s edge to output $\hat{o}_j$ resides on page $\hat{q}_j$. Suppose, with no loss of generality, that $u$ lies to the left of $v$ in the book-embedding. We note the following facts about the portion of the embedding that we have just described.

(1) Since $u$ and $v$ share the same vertex type, we have both

$$\{p_1, p_2, \cdots, p_k\} = \{q_1, q_2, \cdots, q_k\}$$

and

$$\{\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_l\} = \{\hat{q}_1, \hat{q}_2, \cdots, \hat{q}_l\}.$$

(2) Because edges on the same page cannot cross, we have the following:
- For each $i \in \{1, 2, \cdots, k-1\}$, $q_i \notin \{p_{i+1}, \cdots, p_k\}$;
- For each $i \in \{1, 2, \cdots, l-1\}$, $\hat{p}_i \notin \{\hat{q}_{i+1}, \cdots, \hat{q}_k\}$;
- For each $i \in \{1, 2, \cdots, k\}$ and $j \in \{1, 2, \cdots, k\}$, $q_i \neq \hat{p}_j$.

An easy inductive argument using these facts establishes the following:
- For each $i \in \{1, 2, \cdots, k\}$, $p_i = q_i$;
- For each $i \in \{1, 2, \cdots, l\}$, $\hat{p}_i = \hat{q}_i$;
- For distinct $i, j \in \{1, 2, \cdots, k\}$, $p_i \neq p_j$;
- For distinct $i, j \in \{1, 2, \cdots, l\}$, $\hat{p}_i \neq \hat{p}_j$;
- For $i \in \{1, 2, \cdots, k\}$ and $j \in \{1, 2, \cdots, l\}$, $p_i \neq \hat{p}_j$.

This establishes Claims 2 and 3.

We now use Claims 1–3 to complete the proof.

CLAIM 4. *If some two inputs have the same type in the book-embedding, then no two outputs have the same type.*

By Claim 1, the outputs would have to lie on the same side of the similar-typed inputs; by Claims 2 and 3, the outputs would then have distinct types.

CLAIM 5. *If some two inputs have the same type in the book-embedding, then the (common) type of these inputs differs from the types of all of the outputs.*

Say that inputs $u$ and $v$ have the same type in the book-embedding. Focus on an arbitrary output $w$. By Claim 2, the edges from $w$ to both $u$ and $v$ reside on the same page. By Claim 3, all edges emanating from $u$ (and from $v$) reside on distinct pages. Since $\min(m, n) \geqq 2$, this establishes the Claim.

In summary, we have shown the following:
- Either all of the input vertices of $K_{m,n}$, or all of its output vertices, must have distinct vertex types in the book-embedding;
- If some two input (respectively, output) vertices of $K_{m,n}$ have the same type in the embedding, then their common type must differ from the types of all of the output (respectively, input) vertices.

It follows that there must be at least $1 + \min(m, n)$ distinct vertex types in the book-embedding. Moreover,
- The only way to achieve this minimum typenumber is as follows:
  —To place the more numerous of the inputs or outputs in a contiguous block;
  —To dedicate a distinct page to (all of the edges incident to) each of the less numerous of the inputs or outputs.

This last item determines the type-minimizing book-embedding completely, up to rotation. (When $m = n$, we should also add the phrase "up to isomorphism.") □

**Ladders.** For integer $h \geqq 1$, the *height-$h$ ladder graph $L_h$* has vertex-set

$$\{a_1, a_2, \cdots, a_h, b_1, b_2, \cdots, b_h\}$$

and edges connecting the following:
- Each pair $(a_i, b_i)$ for $i \in \{1, 2, \cdots, h\}$;
- Each pair $(a_i, a_{i+1})$ and each pair $(b_i, b_{i+1})$ for $i \in \{1, 2, \cdots, h-1\}$.

PROPOSITION 9. *Restricting attention to* 1-*page book-embeddings of* $L_h$ *we have:* Typenumber ($L_1$) = 2; Typenumber ($L_2$) = 3; Typenumber ($L_3$) = 4; *for* $h \geqq 4$, Typenumber ($L_h$) = 5.

*Proof.* The result is trivial for $h < 3$: $L_1$ is the 1-ary star, so the result is a special case of Proposition 5; $L_2$ is the four-cycle, so its unique 1-page book-embedding has three vertex types, which must be minimal since $L_2$ is not a star.

Focus henceforth on the case $h \geqq 3$.

Consider the 1-page book-embedding of $L_h$ that orders the vertices

$$a_2 - a_3 - \cdots - a_h - b_h - \cdots - b_2 - b_1 - a_1 .$$

When $h = 3$, this embedding uses four vertex types:

$$\{(0,3),(1,1),(2,0),(2,1)\}.$$

When $h = 4$, this embedding uses five vertex types:

$$\{(0,3),(1,1),(1,2),(2,0),(2,1)\};$$

the rotations of this book-embedding also use five vertex types (but, different ones). This establishes the upper bound.

To see the lower bound, note first that $L_h$ is a biconnected outerplanar graph; hence by Lemma 1, a 1-page embedding of $L_h$ has precisely one source and one sink. Since $L_h$ has four bivalent vertices, at least two of these must have type (1, 1).

Consider first the case $h = 3$. We have accounted for three types thus far, the type (1, 1), the source and the sink. We consider three cases.

(1) If the source and sink are both bivalent vertices, then we need at least one more vertex type to account for the two trivalent vertices in $L_3$.

(2) If the source and the sink are both trivalent, then we cannot have a legal book-embedding, for we cannot have on a single page two vertex-disjoint length-four paths with the same endpoints.

(3) If one of the source and sink is bivalent while the other is trivalent, then we need a fourth vertex type to account for the second trivalent vertex.

In all realizable cases, then, we need four vertex types.

When $h \geqq 4$, we need one more ingredient for our argument. Let $n(L, R)$ denote the number of vertices in the embedding that have type $(L, R)$. Since across the entire embedding, the number of edges "leaving some vertex to the right" must equal the number of edges "leaving some vertex to the left," we must have the equation

$$2n(0,2)+3n(0,3)+n(1,2)=2n(2,0)+3n(3,0)+n(2,1).$$

Since we now have at least four trivalent vertices, this conservation equation must be satisfied subject to the following equalities:

- $n(0, 2) + n(1, 1) + n(2, 0) = 4$;
- $n(0, 3) + n(1, 2) + n(2, 1) + n(3, 0) = 2h - 4$;
- $n(0, 2) + n(0, 3) = n(2, 0) + n(3, 0) = 1$.

We easily show that this system of equalities can be satisfied only if

$$n(1,2) \cdot n(2,1) > 0.$$

This means, however, that we have at least five vertex types in the book-embedding: the three two-sided types plus the source and the sink. The lower bound follows. □

We conjecture that additional pages for $L_h$, $h \geqq 4$, will not lower its typenumber below five, but we suspect that only an unilluminating case analysis will settle the conjecture.

**3. Typenumber and graph structure.** We have tried to link the typenumber of a graph with some other structural property of the graph. With the exception of a nontrivial bound connecting the number of pages and the number of types in a book-embedding, our quest has been unsuccessful.

**3.1. Book-thickness.** We begin with our one success. Since trees, which are 1-page embeddable, can have arbitrarily large sets of distinct vertex-degrees, and hence arbitrarily large typenumbers, we cannot hope to bound the book-thickness of $G$ from below using the typenumber of $G$. But, we can obtain a nontrivial upper bound on the book-thickness of $G$ from the typenumber.

PROPOSITION 10. *If the graph $G$ admits a $p$-page book-embedding with $t$ vertex types, then*

$$p \leqq \binom{t}{2};$$

*no smaller bound is generally possible, since this bound is achievable.*

*Proof.* Consider an arbitrary $p$-page $t$-type book-embedding. Say that page $k$ $(1 \leqq k \leqq p)$ is *introduced* by the leftmost vertex $v$ in the embedding that has an edge entering it from the left on page $k$, i.e., that has a positive entry $L_k$ in its vertex type $\tau(v)$ as in (1). Assume that vertex $v$ of $G$ introduces both page $i$ and page $j$ in a book-embedding. Then the vertices $u_1$ and $u_2$ that lie to the left of $v$ in the embedding and that "emit" the witnessing edges on pages $i$ and $j$ must have distinct vertex types: If they had the same type, then the fact that edges on a given page of a book-embedding do not cross would force $v$ to accept both of these edges from the rightmore of $u_1$ and $u_2$; these "parallel edges" would contradict the simplicity of $G$. It follows that a vertex in the embedding can introduce no more pages than it has distinct vertex types to its left in the embedding. Furthermore, if $v$ introduces a page, then its type differs from the types of all the vertices to its left. It follows that the $m$th vertex from the left in the book-embedding can introduce no more than $m - 1$ pages (and this many only if all vertices to its left have distinct types). The bound follows.

To see that the bound is achievable, consider $K_n$. We saw in Proposition 7 that every book-embedding of $K_n$ has $n$ vertex types. Consider the (thickness-inefficient) book-embedding that assigns each edge of $K_n$ to a distinct page. This embedding uses $n$ types and $\binom{n}{2}$ pages. □

Despite whatever hopes our bound might raise, we find that we cannot generally hope to move toward optimizing either typenumber or book-thickness by moving toward optimizing the other.

PROPOSITION 11. (a) *There exist graphs $G$ and $t$-type $p$-page book-embeddings of $G$ such that every $(p + 1)$-page book-embedding of $G$ has more than $t$ types.*

(b) *There exist graphs $G$ such that every $p$-page book-embedding of $G$ uses at least $t + 1$ types, but there is a $(p + 1)$-page book-embedding of $G$ that uses fewer than $t$ types.*

*Sketch of proof.* (a) There is a 1-page two-type book-embedding of the $d$-ary star (Proposition 5). For $d > 1$, any $(p > 1)$-page book-embedding must have more than two types, since some leaves must reside on distinct pages, and hence have distinct types.

(b) By Proposition 8, the book-embedding that minimizes the typenumber of $K_{n,n}$ is unique, up to rotation. This $(n + 1)$-type embedding uses $n + 1$ pages. Bernhart and Kainen [1] exhibit an $n$-page book-embedding of $K_{n,n}$. It follows that type-minimality cannot be achieved simultaneously with thickness-minimality. □

**3.2. Graph homeomorphism.** In the case of graph homeomorphism, not only can we show that taking homeomorphs can either increase or decrease typenumber, we can show that no functional bound can be placed on the amount of change.

PROPOSITION 12. (a) *There exist graphs $G$ and $t$-type book-embeddings of $G$ such that every nontrivial homeomorph of $G$ admits only book-embeddings that use more than $t$ types.*

(b) *Let $D_G$ denote the set of distinct vertex-degrees in the graph $G$. There exists a homeomorph of $G$ that admits a 3-page book-embedding with $|D_G| + 14$ vertex types. Thus, there exist graphs $G$ such that every book-embedding of $G$ uses at least $t$ types, but there is a homeomorph of $G$, one of whose book-embeddings uses strictly fewer than $t$ types.*

*Sketch of proof.* (a) We invoke the $d$-ary star once more. Any nontrivial homeomorphism of the star is no longer a star, and hence cannot be realized with a 2-type book-embedding.

(b) We modify a device of Bernhart and Kainen [1, Thm. 5.4]. Let $G = (V, E)$ be a simple undirected graph. We construct a 3-page-embeddable homeomorph of $G$ in stages, by constructing the book-embedding directly:

(1) Lay the $|V|$ vertices of $G$ along the spine of the book, in arbitrary order.

(2) Place $2|E|$ new vertices along the spine, to the right of $G$'s vertices.

(3) Use one page to lay the edges of a forest of stars, with $G$'s vertices as roots and the new vertices as leaves: each $d$-valent vertex of $G$ becomes the root of a $d$-ary star. In order to fit on one page of the book, these stars must be nested in the sense that $G$'s rightmost vertex uses the leftmost new vertices, $G$'s second rightmost vertex uses the new vertices just to the right of these leftmost ones, and so on.

(4) Using one "pseudopage"—a page that allows crossed edges—connect up the new vertices to construct a homeomorph of $G$, in fact one that has precisely two new vertices along each edge of $G$.

(5) Place a pseudovertex at each edge-crossing on the pseudopage. Distort the edges on the pseudopage so that all of the pseudovertices lie along a line. Place a new pseudovertex at each point where an edge crosses the line.

(6) Pull the line of pseudovertices around 180° rigidly, so that the pseudovertices lie on the spine, to the right of the new vertices.

(7) Add new pseudovertices along the spine to the right of the already-present pseudovertices, wherever an edge crosses the spine.

We now have a 2-page book-embedding of a graph derived from $G$. At this point, we can apply the techniques of Theorem 5.4 of [1] to use the third page to modify this derived graph so that it becomes a homeomorph of $G$. (We refer the reader to that paper rather than repeat the construction.)

The 3-page book-embedded homeomorph has the following characteristics.

- The leftmost $|V|$ vertices have $|D_G|$ different valences; every degree-$d$ vertex in the block has type

$$\begin{pmatrix} 0 & d \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

- All other vertices in the homeomorph are bivalent (as they must be); their types are arbitrary, except that they cannot contain any entry "2," nor can they contain the pattern "(1, 1)" on page three.

The result follows by adding up the number of types. □

## 4. On minimizing typenumber.

**4.1. The infeasibility of minimizing graph-typenumber.** We now show that the problem of finding a typenumber-minimal book-embedding of a given graph is likely to be computationally infeasible. We accomplish this by considering the following decision problem.

**MIN-OUTERPLANAR-TYPES.**

*Instance*: An outerplanar graph $G$ and an integer $k$.

*Question*: Does $G$ have a 1-page embedding using at most $k$ vertex types?

THEOREM 1. *The problem* MIN-OUTERPLANAR-TYPES *is* NP-*complete*.

*Proof.* It is transparent that **MIN-OUTERPLANAR-TYPES** is in NP, so we shall concentrate only on showing that it is NP-hard. To this end, we shall begin by showing that the following problem, **MIN-UNION**, is polynomial-time reducible to **MIN-OUTERPLANAR-TYPES.** We shall then complete the proof by showing that **MIN-UNION** is NP-complete.

**MIN-UNION.**

*Instance*: $m$ pairs $(A_i^1, A_i^2)$ of subsets of $\{1, 2, \cdots, n\}$ and an integer $k$.

*Question*: Are there choices $j_i \in \{1, 2\}$, $1 \leqq i \leqq n$, such that $|\bigcup_i A_i^{j_i}| \leqq k$?

In the following, we specify outerplanar graphs, and portions of them, by presenting a sequence of vertex types; Proposition 2 justifies this mode of specification. The notation does not, of course, imply that the given order is the one used in embedding the graph.

Let an instance of **MIN-UNION** be given. Let $s(i)$ be the vertex sequence[2] $(1, 2)^{i+2}(1, 1)(i + 3, 1)$, and let $\hat{s}(i)$ be the reverse sequence $(1, i + 3)(1, 1)(2, 1)^{i+2}$. To each pair

$$(A_i^1 = \{j_1^1, j_2^1, \cdots, j_{k_i^1}^1\}, A_i^2 = \{j_1^2, j_2^2, \cdots, j_{k_i^2}^2\})$$

associate two copies of the biconnected component

$$B_i = (0,2),(1,1),s(j_1^1+3),s(j_2^1+3), \cdots, s(j_{k_i^1}^1+3), \hat{s}(j_1^2+3),$$

$$\hat{s}(j_2^2+3), \cdots, \hat{s}(j_{k_i^2}^2+3),(2,0). \tag{4}$$

In addition take two copies of

$$E_1 = (0,2),(1,1),\hat{s}(4),\hat{s}(5), \cdots, \hat{s}(n+3),(2,0). \tag{5}$$

If $n > 2m - 2$, take $\lceil (n - 2m + 2)/2 \rceil$ copies of $(0, 2)$, $(1, 1)$, $(2, 0)$. Join all of the above components into one component by identifying the first vertex of every sequence; call the resulting vertex, with degree at least $n + 4$, the *anchor* vertex. Finally, add the two connected components

$$E_2 = (0,2),(1,1),(2,0) \tag{6}$$

and

$$E_3 = (0,2),(1,2),(1,1),(2,1),(1,2),(1,1),(2,1),(2,0). \tag{7}$$

The preceding construction results in an outerplanar graph $G$ that has a 1-page embedding using $k + n + 6$ types if and only if there is a choice of sets from each pair $(A_i^1, A_i^2)$ with a union of size $k$, as we show now.

Suppose that there is a vector of choices of sets, $\bar{l} = (l_1, l_2, \cdots, l_m) \in \{1, 2\}^m$, satisfying $|\bigcup_i A_i^{l_i}| \leqq k$. Then $G$ can be embedded using $k + n + 6$ types, as follows. The components $E_1$, $E_2$, and $E_3$ are embedded as specified in (5), (6), (7) above. If $l_i = 1$, then both copies of $B_i$ are embedded in the order specified in (4) above. If $l_i = 2$, then

[2] $(a, b)^c$ denotes a string of $c$ occurrences of the pair $(a, b)$.

both copies of $B_i$ are embedded in the *reverse* of the order specified in (4). We easily verify that the embedding always uses types

$$\{(1,4),(1,5), \cdots ,(1,n+3),(2,0),(1,1),(0,2),(1,2),(2,1)\}$$

for $E_1$, $E_2$, and $E_3$, and some type $(a_1, a_2)$ where $a_1 + a_2 \geqq n + 4$ for the anchor vertex. Let $\bigcup_i A_i^{l_i} = \{r_1, r_2, \cdots , r_k\}$. Then the only additional types from the $B_i$ are $\{(r_i + 3, 1) : 1 \leqq i \leqq k\}$. Hence the total number of types is $k + n + 6$. (Note that the embedding of the component $E_1$ "swallows up" the types corresponding to the elements $A_j^i$ that are not selected by the vector $\bar{l}$.)

Next, suppose that $G$ can be embedded in one page using at most $k + n + 6$ types. Because each component of $G$ is biconnected, the possible one-page embeddings $G$ are very restricted, as the following lemma indicates.

LEMMA 2 [14]. *A biconnected outerplanar graph has a unique outerplanar embedding, up to rotation and reversal.*

For any embedding of $G$, the components $E_2$ and $E_3$ must use types $(0, 2)$, $(2, 0)$, $(1, 1)$, $(1, 2)$, and $(2, 1)$. In addition, the anchor vertex is of some type $(a_1, a_2)$ where $a_1 + a_2 \geqq n + 4$. Of the two copies of $E_1$, at least one must have the anchor vertex at one end. Hence, we may assume without loss of generality that types $\{(1, i + 3) : 1 \leqq i \leqq n\}$ are used also.

For $1 \leqq i \leqq n$, one copy of $B_i$ must have the anchor vertex at one end. That copy must use either types $\{(j_i^1, 1) : 1 \leqq i \leqq k_i^1\}$ or $\{(j_i^2, 1) : 1 \leqq i \leqq k_i^2\}$. In either case, $|\bigcup_i A_i^{l_i}| \leqq k$ for the corresponding selection vector $\bar{l}$.

We have thus reduced the **MIN-UNION** problem to the **MIN-OUTERPLANAR-TYPES** problem. We now complete our proof by establishing the NP-completeness of **MIN-UNION.** As before, membership in NP is transparent, so we shall show only that the problem is NP-hard, by reducing the well-known NP-complete problem **CLIQUE** [6] to it.

**CLIQUE.**

*Instance*: An undirected graph $G = (V, E)$ and an integer $k$.

*Question*: Does $G$ have a clique of size at least $k$?

Assume, without loss of generality, that $V = \{1, 2, \cdots , n\}$. For $1 \leqq i \leqq n$, let $A_i^1 = \{i\}$ and $A_i^2 = \{j : (j, i) \notin E\}$. Given a selection vector $\bar{l} = (l_1, l_2, \cdots , l_m) \in \{1, 2\}^m$, define $V_1^{\bar{l}} = \{i \in V : l_i = 1\}$, $V_2^{\bar{l}} = \{i \in V : l_i = 2\}$, and $A^{\bar{l}} = \bigcup_i A_i^{l_i}$.

We claim that $G$ has a clique of size $k$ if and only if there is a selection vector $\bar{l}$ such that $|A^{\bar{l}}| \leqq n - k$.

Suppose first that the graph $G$ has a clique $C \subseteq V$. Let $l_i = 2$ for $i \in C$ and $l_i = 1$ for $i \notin C$. Then $A^{\bar{l}} \subseteq V - C$; hence, if $C$ has size $k$, then $|A^{\bar{l}}| \leqq n - k$.

Conversely, if $|A^{\bar{l}}| \leqq n - k$, then the set $V_2^{\bar{l}}$ contains a clique of size $k$. This can be seen as follows. Each $i \in V_1^{\bar{l}}$ appears in $A^{\bar{l}}$, and hence adds one to $|A^{\bar{l}}|$. A given $j \in V_2^{\bar{l}}$ fails to appear in $A^{\bar{l}}$ if and only if it is adjacent to every other vertex of $V_2^{\bar{l}}$. In particular, some $k$ vertices from $V_2^{\bar{l}}$ fail to appear in $A^{\bar{l}}$ if and only if they are all mutually adjacent and hence form a clique.

The argument in the preceding paragraphs establishes that **CLIQUE** is reducible to **MIN-UNION,** so **MIN-UNION** is NP-complete. This completes the proof of the Theorem. □

As an added point of interest, the **MIN-UNION** problem is closely related to the following variation of the **SATISFIABILITY** problem. Given an instance $(A_i^1, A_i^2)$ of **MIN-UNION,** construct a CNF formula with clauses

$$C_j = \{x_i : i \in A_j^1\} \cup \{\neg x_i : i \in A_j^2\}.$$

There is a selection vector $\bar{I}$ for which the union set $A^{\bar{I}}$ has cardinality at most $k$ if and only if there is an assignment of truth values to the $x_i$ that satisfies at most $k$ clauses. This **MIN-CNF-SAT** problem contrasts with the **MAX-2CNF-SAT** problem, which is proved NP-complete in [7].

**4.2. The importance of minimizing embedding-typenumber.** Of the three measures of the cost of a book-embedding, typenumber is the only one that is very sensitive to rotation (or, cyclic permutation). It is obvious that rotation has no effect on the book-thickness of a book-embedding. It is not difficult to verify that rotation can increase (or decrease) the pagewidth of a book-embedding by at most a factor of two; this observation is due to Heath (and is used in [4]). However, there is no a priori bound on how much the typenumber of a book-embedding can be affected by rotation.

To verify this last claim, focus on the (1-page) preorder book-embedding of the height-$h$ complete $d$-ary tree $T_{d,h}$. We have shown in Proposition 6(b) that this embedding never uses more than three vertex types. Moderating this good news (and emphasizing the intended message of this section) is the fact that no nontrivial rotation of the preorder book-embedding has typenumber less than four: any rotation uses at least three types for the nonroot nodes of the tree. Even worse, a positive fraction of the possible rotations of the preorder book-embedding actually use the pessimal number $d + 5$ of types. We now verify this latter assertion.

Consider the following familiar labeling of the nodes of $T_{d,h}$ with strings. Let the root node of $T_{d,h}$ be labeled by the null string, and inductively let the $d$ children of the node labeled by the string $\nu$ be labeled by the strings $\nu 1, \nu 2, \cdots, \nu d$, in left-to-right order of their appearance in the preorder book-embedding. It will be useful to determine how many of these string labels contain all of the "letters" in the set $\{1, 2, \cdots, d-1\}$.

LEMMA 3. *Let $A = \{1, 2, \cdots, d\}$. The number of length-n strings of the form $\alpha x$, where $\alpha \in A$ and where $x$ is a length-$(n-1)$ string that contains all of the letters $\{1, 2, \cdots, d-1\}$ in some order, is precisely*

$$S(n;d) = \sum_{i=0}^{d-1} (-1)^i \binom{d-1}{i} \cdot d \cdot (d-i)^{n-1}.$$

*Proof.* We invoke the Principle of Inclusion and Exclusion [11, Chap. 3]: There are $d^n$ length-$n$ strings over the alphabet $A$; there are $d-1$ properties $P_j(x) \equiv [j$ does not appear in $x]$; for each $1 \leqq i \leqq d-1$,

$$\binom{d-1}{i} \cdot d \cdot (d-i)^{n-1}$$

of these length-$n$ strings have the form $\alpha x$, where $\alpha \in A$ and where $x$ lacks at least $i$ of the letters $\{1, 2, \cdots, d-1\}$. □

We now verify that certain leaves whose labels are close to the form enumerated in Lemma 2 lead to typenumber-pessimal rotations of the preorder book-embedding of $T_{d,h}$, while arbitrary nodes whose labels have that form lead to rotations that are close to pessimal.

PROPOSITION 13. *Consider the preorder book-embedding of $T_{d,h}$, where $h > d \geqq 2$. If we rotate the embedding so that the leftmost node, called the pivot node, is one of the $S(h-2; d)$ leaves whose label has the form $\alpha x 1$, where $\alpha \in \{1, 2, \cdots, d\}$ and where $x$ is a length-$(h-3)$ string that contains all of the letters $\{1, 2, \cdots, d-1\}$ in some order, then the resulting book-embedding has $d+5$ vertex types. For fixed d, as h grows without bound, these $S(h-2; d)$ leaves approach the fraction $(d-1)/d^2$ of the nodes of $T_{d,h}$.*

*Sketch of proof.* Consider the preorder book-embedding, rotated as prescribed in the statement of Proposition 13. Note that when the pivot node $\nu$ has a label of the prescribed form, then in the preorder book-embedding it has, for each $i \in \{0, 1, 2, \cdots, d-1\}$, a node to its left that has precisely $i$ children to $\nu$'s right. The edges to these children change from rightgoing to leftgoing after the prescribed rotation. Using this fact, we can verify that in the prescribed rotation of the preorder embedding: the root of $T_{d,h}$ has type $(k, d-k)$ for some $0 \leqq k \leqq d$; the pivot leaf has type $(0, 1)$, while all leaves that were to its left before the rotation have type $(1, 0)$; all internal nodes that were to the left of the pivot leaf and whose children were also to the left of the pivot leaf before the rotation have type $(1, d)$; the father of the pivot leaf has type $(d+1, 0)$; each internal node that lay to the left of the pivot leaf before the rotation but had $i$ $(i \in \{1, 2, \cdots, d-1\})$ children to the right of the pivot leaf has type $(i+1, d-i)$, while its $i$ children have type $(0, d+1)$. Summarizing, we find the following types:

$$\begin{array}{lll} & (k, d-k) & \text{the root,} \\ & (i, d-i+1) & 0 \leqq i \leqq d+1 \text{ the internal nodes,} \\ (8) & (0,1) & \text{the pivot leaf,} \\ & (1,0) & \text{all other leaves,} \end{array}$$

which are $d+5$ in number.

Finally, note that the claimed cardinality of the set of bad pivot leaves follows immediately from Lemma 2. □

In fact, the situation is even worse than Proposition 13 suggests: Asymptotically as we consider successively taller complete $d$-ary trees, rotating the preorder book-embedding to a random site will, with probability approaching one, yield an embedding with at least $d+4$ vertex types. This follows from the fact that the overwhelming majority of the nodes of $T_{d,h}$ have labels of the form prescribed in Lemma 2, and each is a bad pivot node.

PROPOSITION 14. *Consider the preorder book-embedding of $T_{d,h}$, where $h > d \geqq 2$. If we rotate the embedding so that the pivot node has a label of the form $\alpha x$, where $\alpha \in \{1, 2, \cdots, d\}$ and where $x$ is a string that contains all of the letters $\{1, 2, \cdots, d-1\}$ in some order, then the resulting book-embedding has at least $d+4$ vertex types. For fixed $d$, as $h$ grows without bound, these "bad" pivot nodes constitute the fraction*

$$1 - O\left(\left(\frac{d-1}{d}\right)\right)$$

*of the nodes of $T_{d,h}$.*

*Sketch of proof.* Each length-$n$ string referred to in Lemma 2 is the label of some node at level $n$ of $T_{d,h}$. By the reasoning in the proof of Proposition 13, the typenumber of the preorder book-embedding of $T_{d,h}$, rotated so that any such node is the pivot node, is at least $d+4$: all of the types enumerated in (8), save perhaps $(0, 1)$, must occur. We can, therefore, use Lemma 2 to calculate a lower bound on the number of nodes of $T_{d,h}$ whose labels guarantee such large typenumber. By our earlier reasoning, this number is no less than

$$\sum_{n=d+1}^{h-1} S(n;d) = \sum_{n=d+1}^{h-1} \sum_{i=0}^{d-1} (-1)^i \binom{d-1}{i} \cdot d \cdot (d-i)^{n-1}$$

$$= d \cdot \sum_{i=0}^{d-1} (-1)^i \binom{d-1}{i} \sum_{n=d+1}^{h-1} (d-i)^{n-1}$$

$$= d \cdot \sum_{i=0}^{d-2} (-1)^i \binom{d-1}{i} \cdot \left( \frac{(d-i)^{h-1} - (d-i)^d}{d-i-1} \right) + (-1)^{d-1} \cdot d \cdot (h-d-1).$$

Since $T_{d,h}$ has $(d^h - 1)/(d - 1)$ nodes, the claimed fraction of the pivot nodes follows. □

To give the reader a feeling for how fast the "eventually" of the big-$O$ in Proposition 14 eventuates, we present Table 1 that indicates, as a function of $d$ and $h$, the fraction of potential pivot nodes in $T_{d,h}$ whose labels satisfy the conditions of the proposition; all lead to book-embeddings with typenumber at least $d + 4$.

Propositions 13 and 14, being asymptotic, are at most suggestive in impact. We now present Tables 2 and 3 that indicate the maximum, minimum, and average typenumber observed in a series of experiments that looked at all rotations of a variety of book-embeddings. We rotated the preorder book-embedding of $T_{d,h}$, for a variety of values of $d$ and $h$, and we rotated the (thickness- and width-optimal) 2-page book-embedding from [3] of the height-$h$ $X$-tree $X(h)$, for a few values of $h$. The *height-$h$ $X$-tree* is obtained from the height-$h$ complete binary tree $T_{2,h}$ by adding edges linking all nodes at each level in a line. For perspective, we present an analogue of Proposition 6 for $X$-trees.

PROPOSITION 15. *A 2-page book-embedding of a height-h X-tree can have typenumber at most*

$$\min(56, 2^{h-1} - 2h + 2) + \min(35, 2h - 4) + \min(20, 2^{h-1} - 2) + 3.$$

*Sketch of proof.* The proof is immediate from Proposition 4, given that the height-$h$ $X$-tree has $2^{h-1} - 2h + 2$ nodes of degree 5, $2h - 4$ nodes of degree 4, $2^{h-1} - 2$ nodes of degree three, and three nodes of degree two. □

TABLE 1
*Lower bound on the probability that randomly chosen pivot node yields $d + 4$ types.*

| | Arity | | | | | |
|---|---|---|---|---|---|---|
| Height | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 0.400 | — | — | — | — | — |
| 4 | 0.645 | 0.298 | — | — | — | — |
| 5 | 0.794 | 0.511 | 0.176 | — | — | — |
| 6 | 0.882 | 0.664 | 0.330 | 0.092 | — | — |
| 7 | 0.933 | 0.772 | 0.467 | 0.190 | 0.045 | — |
| 8 | 0.963 | 0.846 | 0.584 | 0.296 | 0.103 | 0.021 |
| 9 | 0.979 | 0.897 | 0.680 | 0.401 | 0.175 | 0.052 |
| 10 | 0.989 | 0.931 | 0.755 | 0.498 | 0.256 | 0.097 |
| 11 | 0.994 | 0.954 | 0.814 | 0.585 | 0.339 | 0.154 |
| 12 | 0.997 | 0.969 | 0.860 | 0.659 | 0.421 | 0.218 |
| 13 | 0.998 | 0.979 | 0.894 | 0.722 | 0.498 | 0.286 |
| 14 | 0.999 | 0.986 | 0.920 | 0.775 | 0.569 | 0.355 |
| 15 | 0.999 | 0.991 | 0.940 | 0.818 | 0.632 | 0.423 |
| 16 | 1.000 | 0.994 | 0.955 | 0.853 | 0.687 | 0.487 |
| 17 | 1.000 | 0.996 | 0.966 | 0.882 | 0.735 | 0.547 |
| 18 | 1.000 | 0.997 | 0.975 | 0.905 | 0.776 | 0.602 |
| 19 | 1.000 | 0.998 | 0.981 | 0.924 | 0.812 | 0.652 |
| 20 | 1.000 | 0.999 | 0.986 | 0.939 | 0.842 | 0.696 |

TABLE 2
*Minimum ($m$), maximum ($M$), and average ($a$) number of d-ary tree vertex-types.*

$d$-ary Trees

| | Arity | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | | | 3 | | | 4 | | | 5 | | | 6 | | |
| Height | $m$ | $M$ | $a$ | $m$ | $M$ | $a$ | $m$ | $M$ | $a$ | $m$ | $M$ | $a$ | $m$ | $M$ | $a$ |
| 2 | 2 | 3 | 2.34 | 2 | 3 | 2.50 | 2 | 3 | 2.60 | 2 | 3 | 2.67 | 2 | 3 | 2.71 |
| 3 | 3 | 5 | 4.29 | 3 | 6 | 4.77 | 3 | 6 | 5.05 | 3 | 6 | 5.23 | 3 | 6 | 5.35 |
| 4 | 3 | 7 | 5.27 | 3 | 7 | 5.78 | 3 | 7 | 6.06 | 3 | 7 | 6.24 | 3 | 7 | 6.36 |
| 5 | 3 | 7 | 5.58 | 3 | 8 | 6.20 | 3 | 8 | 6.57 | 3 | 8 | 6.81 | 3 | 8 | 6.98 |
| 6 | 3 | 7 | 5.76 | 3 | 8 | 6.47 | 3 | 9 | 6.93 | 3 | 9 | 7.25 | 3 | 9 | 7.45 |
| 7 | 3 | 7 | 5.87 | 3 | 8 | 6.65 | 3 | 9 | 7.20 | 3 | 10 | 7.60 | 3 | 10 | 7.91 |
| 8 | 3 | 7 | 5.93 | 3 | 8 | 6.77 | 3 | 9 | 7.40 | 3 | 10 | 7.88 | 3 | 11 | 8.26 |
| 9 | 3 | 7 | 5.96 | 3 | 8 | 6.84 | 3 | 9 | 7.55 | 3 | 10 | 8.11 | 3 | 11 | 8.54 |
| 10 | 3 | 7 | 5.98 | 3 | 8 | 6.90 | 3 | 9 | 7.66 | 3 | 10 | 8.26 | — | — | — |
| 11 | 3 | 7 | 5.99 | 3 | 8 | 6.93 | 3 | 9 | 7.75 | — | — | — | — | — | — |
| 12 | 3 | 7 | 5.99 | 3 | 8 | 6.95 | — | — | — | — | — | — | — | — | — |
| 13 | 3 | 7 | 6.00 | — | — | — | — | — | — | — | — | — | — | — | — |

TABLE 3
*Minimum ($m$), maximum ($M$), and average ($a$) number of X-tree vertex-types.*

$X$-trees

| Height | $m$ | $M$ | $a$ |
|---|---|---|---|
| 3 | 6 | 7 | 6.86 |
| 4 | 11 | 14 | 12.47 |
| 5 | 13 | 20 | 17.74 |
| 6 | 13 | 25 | 20.35 |
| 7 | 14 | 26 | 22.71 |

Tables 2 and 3 demonstrate that the phenomenon that is perhaps exaggerated by the asymptotics of Propositions 13 and 14 obtains even for very modest size situations, at least for the indicated book-embeddings of both complete trees and $X$-trees.

The results of this section establish the need for the algorithm of the next section.

**4.3. Rotating a book-embedding to minimize typenumber.** We now describe a family of efficient algorithms that find the rotation of a book-embedding that minimizes typenumber. All of the algorithms follow the same strategy; they differ in their data structures, which are tailored to the magnitudes of the book-thickness $p$ of the book-embedding and the maximum vertex-degree $d$ of the embedded graph $G$.

**The Algorithmic Strategy.**

**Input.** A $p$-page book-embedding of an $n$-vertex graph $G$, presented via the associated sequence $\tau_1, \tau_2, \cdots, \tau_n$ of vertex types.

**Output.** A list, with one entry for each vertex $v$ of $G$. The entry for each $v$ comprises

- the number of vertex types in the book-embedding obtained by rotating the input embedding so that $v$ becomes the pivot vertex;

- The *bag* (multiset, with multiplicities) of vertex types that appear in the book-embedding when $v$ is the pivot vertex.

**The Strategy.** Say that vertex $v$ is the current pivot vertex of the book-embedding. In order to rotate the embedding, we do the following.

**Step 1.** Rotate $v$'s $p \times 2$ type-matrix so that each row $(0, R_k)$ becomes $(R_k, 0)$; this effectively moves $v$ to the right end of the embedding.

**Step 2.** For each vertex $w$ that is adjacent to $v$, if the edge of adjacency resides on page $k$, then row $k$ of $w$'s type-matrix is changed from $(L_k, R_k)$ to $(L_k - 1, R_k + 1)$; this effectively "swings that edge around."

**Step 3.** As each vertex type is altered in **Steps 1** or **2**, the old type's multiplicity is *diminished* in the bag and the new type's multiplicity is *augmented* in the bag; when a type disappears from the bag, or appears for the first time (detected by the multiplicities), the typecount of $v$ is adjusted accordingly.

**Implementing the strategy.**

We need to specify the data structures used to implement the strategy.

(I) *The Paged Adjacency Table*. It is convenient to make use of a *paged adjacency table* for $G$, i.e., an adjacency list with a record of which edges lie on which pages. Such a list is readily constructed from the input. Two organizations for the list recommend themselves, the choice being dictated by the magnitudes of $p$ (the book-thickness of the input embedding) and of the number of edges of the input graph $G$. In both representations, there is a one-dimensional array (the table) whose entries are the vertices of $G$, in the order they lie along the spine of the book. The table entry corresponding to vertex $v$ of $G$ comprises the following:

- A register *Type–Count*$(v)$ recording the number of vertex types in the book-embedding when $v$ is the pivot vertex;
- A pointer into the bag of vertex types that points, at any given moment, to the then-current vertex type of $v$;
- A pointer to a list of those vertices that are adjacent to $v$ in $G$; each such vertex is represented via a pointer into the table.

The two organizations differ in their representations of the information about which edges lie on which pages.

*Distributed page information*. In this organization, each vertex $w$ in the list of vertices adjacent to vertex $v$ contains a field indicating the page via which $v$ is adjacent to $w$. Since each such field requires $\log_2 p$ bits, and since there is such a field for each endpoint of each of $G$'s $e$ edges, this organization requires

$$2e \cdot \log_2 p$$

bits for recording the page information.

*Centralized page information*. In this organization, the entry of each vertex $v$ in the table contains $p$ subfields; the $i$th subfield contains a pointer to a list of those vertices that are adjacent to $v$ in $G$ via page $i$. This creates $(p - 1)n$ new subfields in all, each of $\log_2 n$ bits, since it must be capable of pointing to any vertex of $G$. Thus, this organization requires

$$(p-1)n \cdot \log_2 n$$

bits for recording the page information.

For all but quite dense graphs, the distributed organization is likely to be the preferred one.

(II) *The Bag of Types*. We must represent the bag of vertex types in a way that facilitates adding, deleting, and altering types, where each alteration either exchanges the $L$- and $R$-entries of a type-matrix (thereby moving the associated vertex from the left end to the right end of the book-embedding), or adds $(-1, +1)$ to some row of the type-matrix (thereby flipping one edge from left-entering to right-entering). As noted earlier, the changes to the bag must maintain an accurate count of the multiplicity of each vertex type in the bag. The preferred representation of a bag depends on the magnitudes of the book-thickness $p$ of the book-embedding and the maximum vertex-degree $d$ of $G$.

(A) *$p$ and $d$ both small*. This case is quite common:

| | | | |
|---|---|---|---|
| binary trees | $p=1$ | $d=3$ | [4], |
| two-dimensional meshes | $p=3$ | $d=4$ | [4], |
| $X$-trees | $p=2$ | $d=5$ | [4], |
| Benes networks | $p=3$ | $d=4$ | [5], |

In this case, (3) assures us that the number of vertex types must be small, and in fact cannot exceed

$$\binom{2p+d}{d}-1;$$

we shall, therefore, represent the bag as a one-dimensional *array* of nonnegative integers, indexed as follows. The array/bag-entry corresponding to vertex type

$$\begin{pmatrix} L_1 & R_1 \\ L_2 & R_2 \\ \vdots & \vdots \\ L_p & R_p \end{pmatrix},$$

each $L_i, R_i \in \{0, 1, \cdots, d\}$, is determined by converting the type-matrix to the length-$2p$ $(d+1)$-ary numeral

$$L_pR_pL_{p-1}R_{p-1}\cdots L_1R_1,$$

and evaluating the numeral; since $p$ is fixed, distinct numerals specify distinct numbers. Each entry in the array is the multiplicity of the vertex type in the book-embedding that indexes that entry.

At any given moment, we shall remember the index of the current pivot vertex $v$ of the book-embedding. To rotate the book-embedding one place, we move the current $v$ to the right end of the embedding, and we use the paged adjacency table to pick up the type of the next pivot vertex (which is the successor in the table of the current pivot vertex). Moving $v$ to the right end of the embedding consists of the following.

**Step 1.** Decrease the multiplicity of $v$'s current vertex type.

**Step 2.** Determine $v$'s new vertex type, which is obtained by multiplying the current type by $d+1$ (recall that all current $L_i$ are zero and that we want to flip each $L_i$ and $R_i$).

**Step 3.** Change the vertex type of each neighbor $w$ of $v$, after decreasing the multiplicity of $w$'s current type; if vertex $w$ is adjacent to $v$ on page $k$ (discovered from the paged adjacency table) then the new vertex type of $w$ is obtained by decreasing the current type by the quantity $d(d+1)^{2k-2}$ (recall that we want to decrease $L_k$ by one and increase $R_k$ by one).

Of course, during these alterations we are keeping track of any vertex types that disappear or appear anew, to evaluate *Type–Count*$(v)$.

(B) *$p$ large and $d$ small.* Although we do not know of any specific graphs with such $p$ and $d$, we do know that they exist: It is proved in [4] that for all $d$ and all sufficiently large $n$, there exist $n$-vertex graphs with maximum degree $d$ that cannot be embedded in fewer than

$$(\text{const.})\frac{n^{1/2-1/d}}{\log^2 n}$$

pages. Since $p$ is large, the array-solution of the previous section is likely to be too wasteful of memory, since the bag/array is likely to be sparsely occupied. We shall, therefore, represent our bag of vertex types as a height-$(2p+1)$ *trie* (digital search tree).

Each internal node of the trie will be a $(d+1)$-ary array of pointers to the adjacent levels of the trie; we assume that all edges are bidirectional. The array positions represent labels on the edges from a node to its children, the labels $0, 1, \cdots, d$ being the valid entries in a vertex type. Each vertex type that occurs in the current rotation of the book-embedding will be a root-to-leaf path in the trie; the multiplicity of the type will be recorded in its leaf node.

At any given moment, we shall point to the leaf-node corresponding to the current vertex type of the current pivot vertex $v$ of the book-embedding. To rotate the book-embedding one place, we move the current pivot $v$ to the right end of the embedding, and we use the paged adjacency table to pick up the type of the next pivot vertex. Moving $v$ to the right end of the embedding consists of the following.

**Step 1.** Decrease the multiplicity of $v$'s current vertex type.

**Step 2.** Determine $v$'s new vertex type, which is obtained by proceeding along the (unique) path from $v$'s leaf to the root of the trie, constructing the vertex type "complementary" to $v$'s, i.e., converting $v$'s current type

$$\begin{pmatrix} 0 & T_1 \\ 0 & T_2 \\ & \vdots \\ 0 & T_p \end{pmatrix}$$

to $v$'s new type

$$\begin{pmatrix} T_1 & 0 \\ T_2 & 0 \\ \vdots & \\ T_p & 0 \end{pmatrix};$$

if the multiplicity of $v$'s current type is decreased to zero in **Step 1,** then, as a space-saving measure, all trie-entries unique to this vertex type can be removed during the leaf-to-root traversal (i.e., all trie-nodes can be removed, up to the occurrence of the first binary node along the path).

**Step 3.** Record $v$'s new vertex type by traversing the root-to-leaf path in the trie dictated by the new vertex type, inserting new nodes when necessary, and increasing this type's multiplicity when the leaf node is reached.

**Step 4.** Change the vertex type of each neighbor $w$ of $v$, after decreasing the multiplicity of $w$'s current type; this is accomplished by traversing the path from the leaf containing $w$'s vertex type toward the root, until we encounter the type-entry for the page on which $v$ is adjacent to $w$ (as in **Step 3**, node entries for a zeroed type can be removed during this traversal); we now reverse direction, following the root-to-leaf path dictated by $w$'s new type, adding new nodes when necessary, and increasing this new type's multiplicity when the leaf node is reached.

(C) *$p$ and $d$ both large*. When both $p$ and $d$ are large, as with the complete graphs $K_n$ or complete bipartite graphs $K_{m,n}$, any data structure that incorporates arrays is likely to be wasteful of space. The trie structure of the previous section deals well with large $p$; with one small modification, it accommodates large $d$ also. Rather than have a $(d+1)$-entry array of pointers at each nonleaf node of the trie, we shall now have a balanced binary tree, having up to $\lceil \log_2 (d+1) \rceil$ levels. The processing of the trie proceeds as in the previous section, with the one complication that, as vertex types are added or deleted, the nonleaf nodes' balanced trees are dynamically updated. Details are left to the reader.

**Analyzing the implementation.** As the algorithm proceeds, the vertex type of a degree-$d$ vertex $v$ of $G$ is changed $d+1$ times: once as $v$ is moved from the left end to the right end of the book-embedding, and once as each of its neighbors is so moved. The structure of the paged adjacency table allows us to determine in time $O(1)$ what vertex type to access, how to access it, and what change to make. The cost of maintaining the bag of vertex types depends on the magnitudes of $p$ and $d$; even with our trie of trees data structure, which is the most costly of the three we describe, each transaction involved in altering a type can be done in time proportional to $p \cdot \log d$. If the graph $G$ has $n$ vertices, $e$ edges, and the set $D$ of distinct vertex-degrees, then the entire algorithm can be executed within time proportional to

$$\sum_{d \in D} p \cdot (d+1) \cdot \log d$$

which is in turn proportional to

$$p \cdot e \cdot \log d_{\max}.$$

**Acknowledgments.** The authors thank Reuven Bar Yehuda, Sandeep Bhatt, and Lenny Heath for helpful and stimulating conversations concerning the material in this paper.

## REFERENCES

[1] F. BERNHART AND P. C. KAINEN, *The book thickness of a graph*, J. Combin. Theory Ser. B, 27 (1979), pp. 320-331.

[2] J. F. BUSS AND P. W. SHOR, *On the pagenumber of planar graphs*, in Proc. 16th Annual ACM Symposium on Theory of Computing, 1984, pp. 98-100.

[3] F. R. K. CHUNG, F. T. LEIGHTON, AND A. L. ROSENBERG, DIOGENES—*A methodology for designing fault-tolerant processor arrays*, in Proc. 13th Internat. Conference on Fault-Tolerant Computing, 1983, pp. 26-32.

[4] ———, *Embedding graphs in books: a layout problem with applications to* VLSI *design*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 33-58.

[5] R. A. GAMES, *Optimal book embeddings of the* FFT *butterfly, Benes, and barrel shifter networks*, Algorithmica, 1 (1986), pp. 233-250.

[6] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman, New York, 1979.

[7] M. R. GAREY, D. S. JOHNSON, AND L. J. STOCKMEYER, *Some simplified* NP-*complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237-267.

[8] L. S. HEATH, *Embedding planar graphs in seven pages*, in Proc. 25th Annual IEEE Symposium on Foundations of Computer Science, 1984, pp. 74–83.

[9] ———, *Embedding outerplanar graphs in small books*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 198–218.

[10] D. J. MUDER, *Book embeddings of regular complete bipartite graphs*, The MITRE Corp., 1985, typescript.

[11] J. RIORDAN, *An Introduction to Combinatorial Analysis*, John Wiley, New York, 1958.

[12] A. L. ROSENBERG, *The* DIOGENES *approach to testable fault-tolerant arrays of processors*, IEEE Trans. Comput., 32 (1983), pp. 902–910.

[13] E. STÖHR, *A tradeoff between pagenumber and width of book embeddings of graphs*, Inform. and Comput., 79 (1988), pp. 155–163.

[14] M. M. SYSLO, *Characterizations of outerplanar graphs*. Discrete Math., 26 (1979), pp. 47–53.

[15] M. YANNAKAKIS, *Four pages are necessary and sufficient for planar graphs*, in Proc. 18th Annual ACM Symposium on Theory of Computing, 1986, pp. 104–108.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 176–196, May 1989

# SCHEDULING TREE-STRUCTURED TASKS ON TWO PROCESSORS TO MINIMIZE SCHEDULE LENGTH*

JIANZHONG DU† AND JOSEPH Y-T. LEUNG†

**Abstract.** Consider a set of $n$ tasks with a tree-structured precedence relation and execution time of 1 or 3 units. We give an $O(n^2 \log n)$-time algorithm to find a minimum length schedule for these tasks on two identical processors. Possible generalization to the case of 1 or $k$ units is also given.

**Key words.** multiprocessor scheduling, schedule length, nonpreemptive scheduling, tree-structured precedence relation

**AMS(MOS) subject classifications.** 90B35, 68R05

**1. Introduction.** A fundamental problem in scheduling theory is that of nonpreemptively scheduling a set $TS = \{T_1, T_2, \cdots, T_n\}$ of $n$ tasks on $m$ identical processors so as to minimize the schedule length. Associated with each task $T_i$ is an execution time $p(T_i)$. The precedence constraints among the tasks is given by a directed acyclic graph $G = (TS, E)$; $T_i$ must finish execution before $T_j$ can start if $(T_i, T_j) \in E$. The problem is to find a minimum length schedule for $TS$, where the length of a schedule is the time taken to execute all tasks in $TS$. Such a schedule will be called an *optimal* schedule.

The complexity of finding an optimal schedule has been studied extensively and a rich collection of results has been obtained. To begin with, if the execution times are arbitrary, then the problem is NP-hard even when $m = 2$ and $G$ has no edges [4], [9]. On the other hand, if the execution times are equal, then the problem is polynomially solvable for arbitrary $m$ when $G$ is a tree [5]. For arbitrary directed acyclic graph, the problem is polynomially solvable when $m = 2$ [1], [3], but it becomes NP-hard if $m$ is arbitrary [8], [9]. For fixed $m > 2$, the complexity of the problem is still open even though a considerable amount of effort has been invested in it. The above results suggest that the complexity of the problem is fairly well understood when the execution times fall into two extremes: arbitrary and equal. Recently, there have been some interest in the complexity of this problem when the execution times are drawn from an arbitrary set with fixed cardinality. In this case the problem is polynomially solvable for arbitrary $m$ when $G$ has no edges [6]. For arbitrary directed acyclic graphs, the problem is NP-hard even when $m = 2$ and $p(T_i) \in \{1, 2\}$ for $1 \leqq i \leqq n$ [8], [9]. When $G$ is a tree, the problem has recently been shown to be NP-hard if $m$ is arbitrary and $p(T_i) \in \{1, k\}$ for $1 \leqq i \leqq n$, where $k$ is arbitrary [2]. It is still an open question whether the problem is polynomially solvable for fixed $m \geqq 2$ and $p(T_i) \in \{1, k\}$ for $1 \leqq i \leqq n$. However, there is an $O(n \log n)$-time algorithm for finding an optimal schedule when $m = 2$ and $p(T_i) \in \{1, 2\}$ for $1 \leqq i \leqq n$ [7].

The purpose of this paper is to shed some light on this open question. The main result consists of an $O(n^2 \log n)$-time algorithm to find an optimal schedule for $m = 2$ and $p(T_i) \in \{1, 3\}$ for $1 \leqq i \leqq n$. As we shall see in the sequel, our algorithm suggests that finding an optimal schedule for $m = 2$ and $p(T_i) \in \{1, k\}$, $1 \leqq i \leqq n$, might be polynomially solvable for each fixed $k$, although the complexity and the difficulty of obtaining such an algorithm increases rapidly with $k$. Our algorithm works in the same spirit as that given in [7]. That is, the tasks are initially scheduled by a special list-

* Received by the editors June 15, 1988; accepted for publication (in revised form) November 23, 1988. This research was supported in part by Office of Naval Research grant N00014-87-K-0833.

† Computer Science Program, University of Texas at Dallas, Richardson, Texas 75083.

scheduling algorithm, also called the Largest-Subtree-First (LSF) Algorithm as in [7], and then the LSF schedule will be improved by one of the several procedures given later if it is not already optimal. However, the procedures to improve the LSF schedule are much more complex and involved than that given in [7].

The organization of the paper is as follows. In the next section we define the Largest-Subtree-First Algorithm and derive conditions under which the LSF schedule is optimal. In § 3 we classify the cases, to be called degenerate cases, for which the LSF schedule might not be optimal. In § 4 we give procedures to improve the LSF schedule for the degenerate cases. Finally, we draw some conclusions in the last section.

**2. The Largest-Subtree-First Algorithm.** In this section we describe the Largest-Subtree-First (LSF) Algorithm and derive conditions under which the LSF schedule is optimal. We also set up common notation that will be used throughout this paper. The task system is given by $G = (TS, E)$, where $G$ is an out-tree rooted at some task $R$ in $TS$ and $TS = \{T_1, T_2, \cdots, T_n\}$ is a set of $n$ tasks with $p(T_i)$ denoting the execution time of $T_i$ for $1 \leqq i \leqq n$. Unless stated otherwise, we assume $p(T_i) \in \{1, 3\}$ for $1 \leqq i \leqq n$. We assume the reader is familiar with certain standard terminologies used in describing a tree-structured task system such as the following: successor, immediate successor, predecessor, immediate predecessor, root, leaf, chain, the length of a chain, two tasks being independent to each other, and the subtree rooted at a task $T$; their definitions can be found in [7].

The total execution time of the tasks in the subtree rooted at a task $T$ is called the *weight* of $T$ and is denoted by $w(T)$. For a given set of tasks $A$, $w(A)$ denotes the total execution time of all tasks in $A$. A task $T$ is called a $j$-unit task if $p(T) = j$. A task $T$ partitions $TS$ into two subsets of tasks—the set of successors of $T$, denoted by *suc* $(T)$, and the set of nonsuccessors of $T$ (including $T$), denoted by *nsc* $(T)$. If $T_i$ is a predecessor of $T_j$, then the *distance* between $T_i$ and $T_j$, denoted by $d(T_i, T_j)$, is defined to be the length of the chain from $T_i$ to $T_j$. $d(T)$ is an abbreviation of $d(R, T)$, where $R$ is the root of the out-tree $G$. The *height* of $T$, denoted by $h(T)$, is the maximum distance from $T$ to a leaf task in $TS$.

For a given schedule $S$ and a task $T$, we use $s(S, T)$ to denote the starting time of $T$ in $S$, $f(S, T)$ to denote the finishing time of $T$ in $S$, and $\sigma(S)$ to denote the length of $S$. $\Psi(S, T)$ denotes the set of tasks that start at or after $f(S, T)$ in $S$. We define the *dual* task of $T$ in $S$ to be the task executing in the interval $[f(S, T) - 1, f(S, T)]$ on the other processor in $S$. *Internal* and *external* idle intervals of a schedule are defined as in [7]. However, we call the first internal idle interval of a schedule the *initial* idle interval. Given two schedules $S_1$ and $S_2$, the concatenation of $S_1$ and $S_2$, denoted by $C(S_1, S_2)$, is the schedule obtained by appending $S_2$ to $S_1$.

Our algorithm will be called Algorithm ONE-THREE and it is mainly based on a list scheduling algorithm called the Largest-Subtree-First (LSF) Algorithm. The LSF Algorithm first orders the tasks in $TS$ according to the LSF ordering, and then the list of tasks is used by a list scheduler to construct a schedule. In the LSF ordering of $TS$, $T_i$ precedes $T_j$ in the list if

(1) $w(T_i) > w(T_j)$, or

(2) $w(T_i) = w(T_j)$ and $p(T_i) < p(T_j)$, or

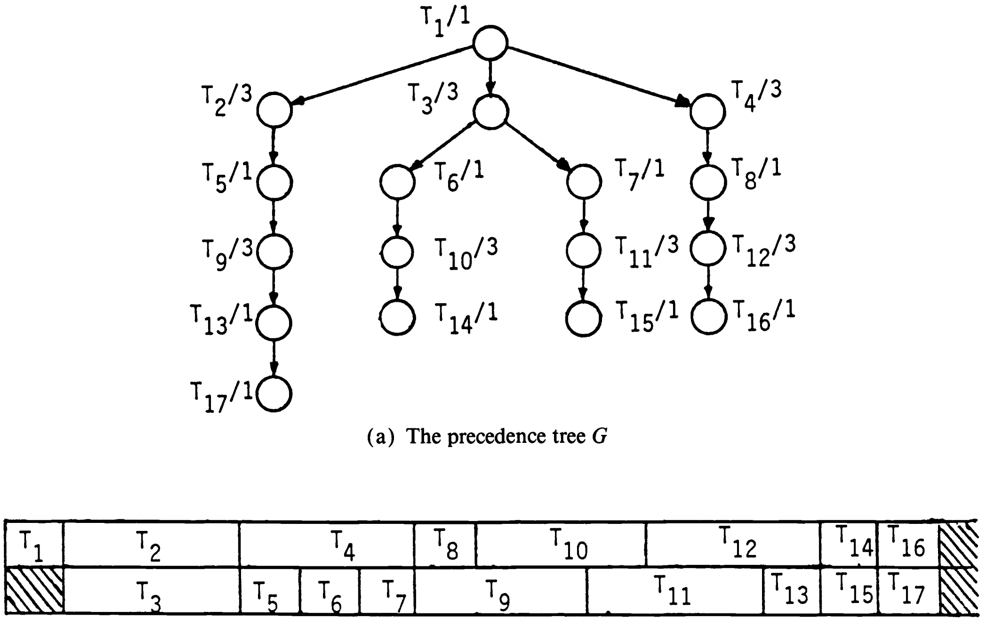
(3) $w(T_i) = w(T_j)$, $p(T_i) = p(T_j)$ and $h(T_i) \geqq h(T_j)$.

Any remaining ties can be resolved arbitrarily. The list schedule constructed by the LSF Algorithm will be called the LSF schedule and is denoted by $S_l$. We use $s(T)$, $f(T)$, $\sigma$, and $\Psi(T)$ to denote $s(S_l, T)$, $f(S_l, T)$, $\sigma(S_l)$, and $\Psi(S_l, T)$, respectively. $S_l^+(T)(S_o^+(T))$, and $S_l(T)(S_o(T))$ denote the LSF (optimal) schedule for the tasks in *nsc*$(T)$ and the tasks in the subtree rooted at $T$, respectively. After eliminating $T$ from $S_l(T)(S_o(T))$, a subschedule of $S_l(T)(S_o(T))$ consisting of the tasks in *suc*$(T)$ is obtained. We denote this subschedule by $S_l^-(T)(S_o^-(T))$. Clearly, $C(S^+, S^-)$ is a valid schedule for the tasks in $TS$, where $S^+ \in \{S_l^+(T), S_o^+(T)\}$ and $S^- \in \{S_l^-(T), S_o^-(T)\}$.

Figure 1 gives a LSF schedule that is optimal; the optimality can be deduced from the fact that there is no idle interval in the schedule except the initial one caused by the root of the out-tree. Figure 2 shows that LSF schedules are not always optimal. Moreover, it may not be easy to decide if a LSF schedule is optimal. For example, the LSF schedule in Fig. 3 has a 2-unit-long external idle interval just like the LSF schedule in Fig. 2. However, the schedule in Fig. 3 is optimal while the one in Fig. 2 is not. As we shall see later, Algorithm ONE-THREE can be viewed as a divide-and-conquer method. It first constructs the LSF schedule $S_l$ for the tasks in $TS$. If $S_l$ is not clearly optimal (to be called degenerate and is defined in § 3), then the algorithm will improve it using the methods described in § 4, until at some point it either discovers that an improvement is impossible or an improved schedule is obtained. We show that if an improvement is impossible, then $S_l$ is already optimal; otherwise, the improved schedule is optimal.

In the remainder of this section we present a fundamental result concerning LSF schedules, and then set up a set of simple conditions under one of which LSF schedules are optimal. Instead of assuming the execution times are of 1 or 3 units, we derive the results for the most general execution times. Lemma 1 only assumes that the maximum execution time is $k$ units.

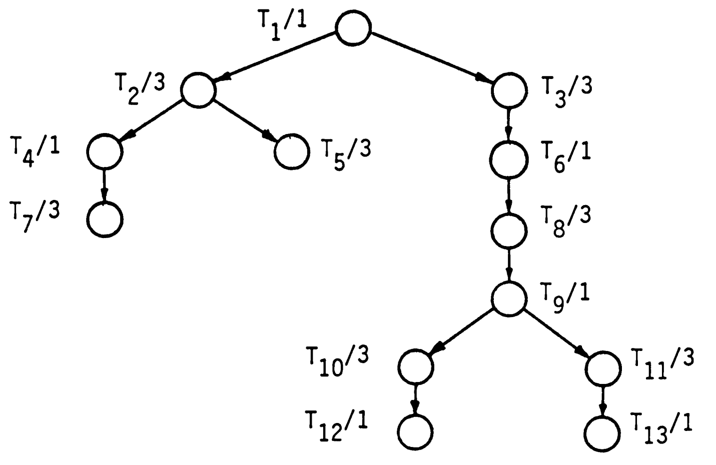
LEMMA 1. *Suppose $T$ is an arbitrary task executing in the interval $[s(T), f(T)]$ in $S_l$. If $[f(T)-1, f(T)]$ is contained in an idle interval of $S_l$, say $[t_1, t_2]$, then let $T'$ be a dummy task such that $f(T') = t_1$ and $w(T') = 0$; otherwise, let $T'$ be any task executing in $[s(T), f(T)]$ in $S_l$. We have $f(T) = d(T)$ if $w(suc(T)) > f(T') - f(T) + \tilde{M} + k$,*



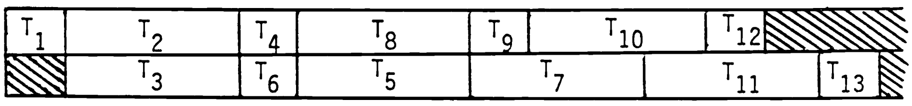
(a) The precedence tree $G$

(b) The LSF schedule $S_l$

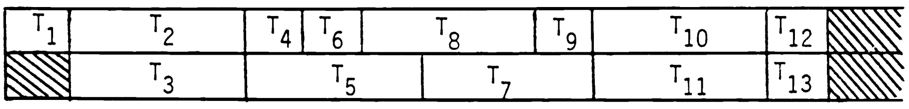
FIG. 1. *An* LSF *schedule that is not degenerate.*



(a) The precedence tree $G$



(b) The LSF schedule $S_l$



(c) The optimal schedule $C(S_o^+(X),\ S_l^-(X))$

FIG. 2. *Case-*1 *degenerate* LSF *schedule* (*improvable*).

*where $\tilde{M}$ is the total weight of all those subtrees that are left unscheduled at $f(T')$ in $S_l$ and have a root independent with $T$.*

*Proof.* Assume the lemma is not true and there is a chain of $j+1$ tasks in $S_l$, $(T_{j+1}, T_j, \cdots, T_1)$, such that $f(T_{j+1}) < s(T_j)$, $f(T_i) = s(T_{i-1})$ for $2 \leqq i \leqq j$ and $T_1 = T$. Since $T_j$ is ready but not executed at $f(T_{j+1})$, both processors must be busy in the interval $[f(T_{j+1}), s(T_j)]$ in $S_l$. Let $\hat{T}_1$ and $\hat{T}_2$ be the two tasks executed in the interval $[s(T_j) - 1, s(T_j)]$. Clearly, $\hat{T}_1$, $\hat{T}_2$, and $T_j$ are independent of one another, and the successors of $\hat{T}_1$ and $\hat{T}_2$ must be executed at or after $s(T_j)$. Since there is a chain executing in $[s(T_j), f(T_1)]$, we have

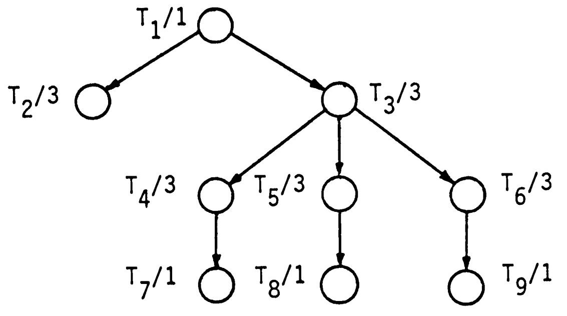
$$w(\hat{T}_1) \leqq f(T_1) - s(T_j) + f(T') - f(T) + \tilde{M} + p(\hat{T}_1), \qquad \text{and}$$

$$w(\hat{T}_2) \leqq f(T_1) - s(T_j) + f(T') - f(T) + \tilde{M} + p(\hat{T}_2).$$

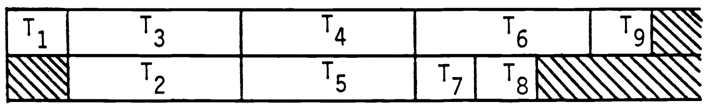
However, $w(T_j) \geqq f(T_1) - s(T_j) + w(suc(T_1)) > f(T_1) - s(T_j) + f(T') - f(T) + \tilde{M} + k$ by the assumption of the lemma. Therefore, we have $w(T_j) > w(\hat{T}_1)$ and $w(T_j) > w(\hat{T}_2)$. This is impossible since the LSF Algorithm would schedule $T_j$ before $\hat{T}_1$ or $\hat{T}_2$. □

COROLLARY 1. *$S_l$ is optimal for equal-execution-time task systems.*

*Proof.* For equal-execution-time task systems we may assume that $k = 1$. We consider two cases of $S_l$—the length of the external idle interval is more than 1 unit, and otherwise. In the former case, we let $T$ be the task such that $f(T) = \sigma$. Applying Lemma 1 to $T$,



(a) The precedence tree $G$



(b) The LSF schedule $S_1$ (also an optimal one)

FIG. 3. *Case-2 degenerate* LSF *schedule* (*U is undefined*).

we have $f(T) = d(T)$ and hence $S_l$ is optimal. In the latter case, we let $[t_1, t_2]$ be the last internal idle interval in $S_l$ and let $T$ be the task such that $f(T) = t_2$. By Lemma 1, we again have $f(T) = d(T)$. Since $\Psi(T) = suc(T)$ and the length of the external idle interval is no more than 1 unit, $S_l$ is clearly optimal. □

From Lemma 1 we can derive several simple conditions under any one of which $S_l$ is optimal. We give these conditions in the following lemma. Lemma 2 assumes that the execution times are 1 or $k$ units.

LEMMA 2. *An* LSF *schedule $S_l$ is optimal if one of the following five conditions holds.* (2.1) $\sigma - p(R) \leqq k$. (2.2) *there is a chain with length $\sigma$.* (2.3) *the length of the external idle interval is more than k units.* (2.4) *the number of idle processor time units in* $[\sigma - k, \sigma]$ *is no more than one.* (2.5) *an internal idle interval intersects* $[\sigma - k - 1, \sigma - k + 1]$.

*Proof.* $S_l$ is clearly optimal if (2.2) holds. If (2.1) holds for $S_l$, then there are two cases to consider: *suc* $(R)$ contains at least one $k$-unit task, and otherwise. In the former case, $S_l$ is optimal since there is a chain of length $\sigma$. In the latter case, all tasks except possibly $R$ have execution time 1 unit. Therefore, *suc* $(R)$ can be considered as an equal-execution-time task system and hence $S_l$ is optimal by Corollary 1. From (2.3) we can infer (2.2) by Lemma 1. Hence, $S_l$ is optimal if (2.3) holds. For (2.4) let $[t_1, t_2]$ be the last internal idle interval that begins earlier than $\sigma - k$ and let $T$ be the task such that $f(T) = t_2$. Since $t_1 < \sigma - k$, we can apply Lemma 1 to $T$ to obtain $d(T) = f(T)$. Noting that $\Psi(T) = suc(T)$ and that there is at most 1 unit of idle processor time after $f(T)$ by (2.4), we conclude that $S_l$ is optimal. For (2.5) let $[t_1, t_2]$ be the internal idle interval that intersects $[\sigma - k - 1, \sigma - k + 1]$ and let $T$ be the task such that $f(T) = t_2$. Since $t_1 < \sigma - k + 1$, we have $f(T) = d(T)$ by Lemma 1. Furthermore, we have $\sigma - t_2 \leqq k$. Since $\Psi(T) = suc(T)$, we have that $\sigma - t_2$ is the optimal schedule length for the tasks in $suc(T)$ by (2.1). Therefore, $S_l$ is optimal. □

If the execution times are 1 or 3 units, then Lemma 2 can be restated as follows.

COROLLARY 2. *An* LSF *schedule $S_l$ is optimal if one of the following three conditions holds*: (C2.1) $\sigma - p(R) \leqq 3$; (C2.2) *there is a chain of length $\sigma$*; *and* (C2.3) *the length of the external idle interval is at most* 1 *unit or at least* 4 *units.*

*Proof.* (C2.1) and (C2.2) follow from (2.1) and (2.2) of Lemma 2, respectively. Combining (2.3), (2.4), and (2.5) of Lemma 2, we obtain (C2.3). □

To conclude this section, we comment on the three criteria used in defining the LSF ordering. The results derived in this section only make use of the property of the first criterion. The other two criteria will be used in a few places in later sections and they are not as important as the first one.

## 3. Classifications of degenerate cases.

In this and the next sections, we assume that the execution times are 1 or 3 units. In the last section we give conditions under any one of which LSF schedules are optimal. In this section we provide better characterizations of LSF schedules that may not be optimal. Our characterization is based on the length of the chain executed at the end of $S_l$, denoted by $L$, and the length of the external idle interval in $S_l$, denoted by $N$. By a chain of length $L$ executed at the end of $S_l$, we mean that there is a chain of length $L$, $(T_j, T_{j-1}, \cdots, T_1)$, such that $s(T_j) = \sigma - L$ and $f(T_1) = \sigma$, and no chain longer than $L$ has that property in $S_l$. We say that $S_l$ is *degenerate* if $\sigma - p(R) > 3$, $2 \leqq N \leqq 3$ and $L < \sigma$. The LSF schedules given in Figs. 2 and 3 are examples of degenerate LSF schedules.

LEMMA 3. *The following statements are true.* (3.1) *$S_l$ is optimal if $S_l$ is not degenerate.* (3.2) *$L \geqq 2$ if $S_l$ is degenerate.* (3.3) *Suppose $S_l$ is degenerate; if $[t_1, t_2]$ is an internal idle interval of $S_l$ and $T$ is the task that finishes at $t_2$, then $\Psi(T) = suc(T)$ and $f(T) = d(T)$.* (3.4) *If $S_l$ is not optimal, then the length of an optimal schedule is $\sigma - 1$.*

*Proof.* Statement (3.1) follows from the definition of $S_l$ being degenerate and Corollary 2. Since the tasks scheduled in the external idle interval of $S_l$ form a chain, (3.2) follows from $N \geqq 2$. For (3.3), since a processor is idle in $[t_1, t_2]$ in $S_l$, the tasks executed after $t_2$ must be successors of $T$. Therefore, $\Psi(T) = suc(T)$. Since $S_l$ is degenerate, we have $t_2 < \sigma - N$. By Lemma 1, we have $f(T) = d(T) = t_2$ and hence (3.3) is proved. For (3.4), $S_l$ must be degenerate since it is not optimal. Let $[\hat{t}_1, \hat{t}_2]$ be the last internal idle interval (or the initial idle interval if there is no internal idle interval) of $S_l$. Then from (3.3), there is a task $Z$ such that $f(Z) = d(Z) = \hat{t}_2$ and $\Psi(Z) = suc(Z)$. Therefore, to improve $S_l$, all we can do is shorten the schedule for the tasks in $suc(Z)$. However, since $2 \leqq N \leqq 3$, the most we can shorten is 1 unit. Thus, (3.4) is true. □

We now classify the degenerate LSF schedules into six cases. The classification is based on the value of $L$: $L > 3$; $L = 3$; and $L = 2$. Note that by Lemma 3, $L \geqq 2$ for a degenerate LSF schedule. We show in Lemmas 4, 5, and 6 that the six cases defined below cover all possible degenerate LSF schedules.

*Case* 1. (A) $L \geqq 4$ and $N = 2$. (B) There is a task $X$ with exactly two immediate successors, *son* 1 and *son* 2, such that $f(X) = d(X)$, $\Psi(X) = suc(X)$, $w(son\ 1) = w(son\ 2) = h(son\ 1) = h(son\ 2) = \sigma - f(X) - 2 = L$ and $p(son\ 1) = p(son\ 2) = 3$. (C) There is a 3-unit leaf task $Y$ such that $f(Y) = f(X) + 2$.

Figure 2 shows a Case-1 degenerate schedule. In this example, $X = T_9$, *son* $1 = T_{10}$, *son* $2 = T_{11}$, and $Y = T_7$, while $L = 4$ and $\sigma = 15$. Note that $T_7$ finishes 2 units later than $T_9$, and the two sons of $T_9$ are the heads of two chains of length 4.

*Case* 2. (A) $L = 4$ and $N = 2$. (B) The tasks scheduled in $[\sigma - 7, \sigma]$ consist of three independent 3-unit tasks, $\hat{T}_1$, $\hat{T}_2$, and $\hat{T}_3$, and three 1-unit tasks, $\hat{U}_1$, $\hat{U}_2$, and $\hat{U}_3$. For each $1 \leqq i \leqq 3$, $\hat{U}_i$ is the only successor of $\hat{T}_i$. Furthermore, $\hat{T}_1$ and $\hat{T}_2$ start at $\sigma - 7$, while $\hat{T}_3$ starts at $\sigma - 4$.

An example of a Case-2 degenerate schedule is shown in Fig. 3. In Fig. 3, $\hat{T}_1$, $\hat{T}_2$, $\hat{T}_3$, $\hat{U}_1$, $\hat{U}_2$, and $\hat{U}_3$ mentioned above are $T_4$, $T_5$, $T_6$, $T_7$, $T_8$, and $T_9$, respectively.

OBSERVATION 1. If $S_l$ is the LSF schedule of the task system $G$ and $S_l$ is Case-2 degenerate, then the length of the longest chain in $G$ is at most $\sigma - 3$.

*Case* 3. (A) $L = 3$ and $N = 3$. (B) The tasks executing in $[\sigma - 6, \sigma]$ consist of three 3-unit leaf tasks.

Figures 4 and 5 give examples of Case-3 degenerate schedules. In the schedule shown in Fig. 4, $T_4$, $T_7$, and $T_8$ are the three 3-unit leaf tasks executed in $[\sigma - 6, \sigma]$.

OBSERVATION 2. If $S_l$ is the LSF schedule of the task system $G$ and $S_l$ is Case-3 degenerate, then the length of the longest chain in $G$ is at most $\sigma - 3$.

*Case* 4. (A) $L = 3$ and $N = 2$. (B) The task that finishes at $\sigma$ is a 3-unit leaf task and the task that finishes at $\sigma - 2$ is a 1-unit leaf task. (C) The two tasks that finish at $\sigma - 3$ are 3-unit tasks, at least one of which is a leaf task.

We always use $\hat{U}$ to denote the 1-unit leaf task that finishes at $\sigma - 2$ in a Case-4 degenerate schedule. An example of a Case-4 degenerate schedule can be found in Fig. 6. In this example, $T_{10}$, $T_8$, and $T_9$ are the 3-unit tasks mentioned in the above definition, and $T_{11}$ is the 1-unit leaf task. Note that $T_8$ could be a leaf task, although it is the immediate predecessor of $T_{11}$ in this case.

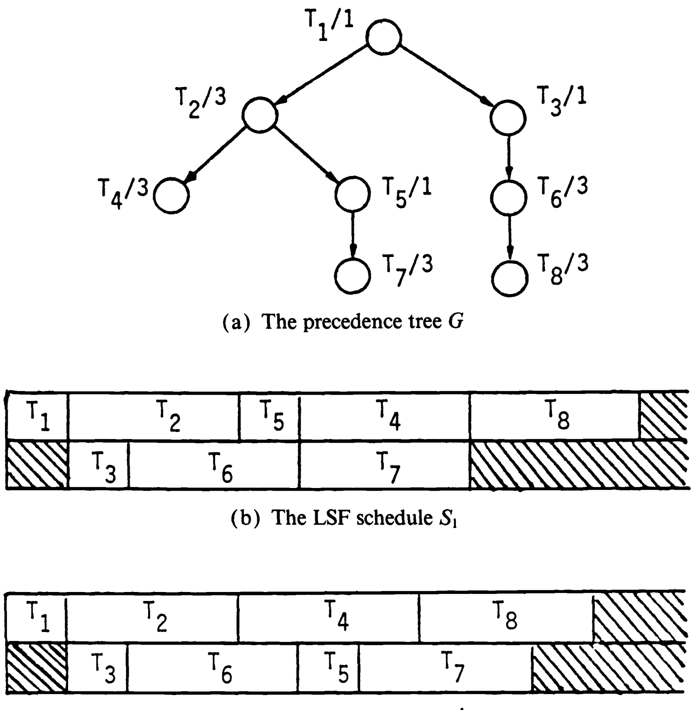
OBSERVATION 3. If $S_l$ is the LSF schedule of the task system $G$ and $S_l$ is Case-4 degenerate, then the length of the longest chain in $G$ is at most $\sigma - 2$.

*Case* 5. (A) $L = 3$ and $N = 2$. (B) There are three 3-unit leaf tasks that start at $\sigma - 3$, $\sigma - 5$, and $\sigma - 6$, respectively.

An example of a Case-5 degenerate schedule is shown in Fig. 7. In this example, $T_5$, $T_6$, and $T_7$ are the three 3-unit leaf tasks mentioned above.

*Case* 6. (A) $L = 2$ and $N = 2$. (B) There is a chain of length 2 that starts at $\sigma - 2$, while the two tasks that finish at $\sigma - 2$ are 3-unit leaf tasks.

The simplest example of a Case-6 degenerate schedule is given in Fig. 8. As can be seen from this example, there is an easy way to improve this degenerate case. That is, in

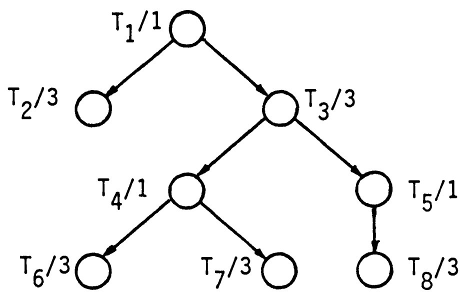


(a) The precedence tree $G$

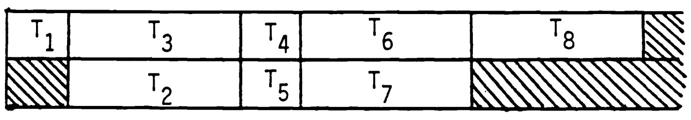
(b) The LSF schedule $S_l$

(c) The optimal schedule $\dot{S}_l$

FIG. 4. *Case-*3 *degenerate* LSF *schedule* (*U can be delayed*).

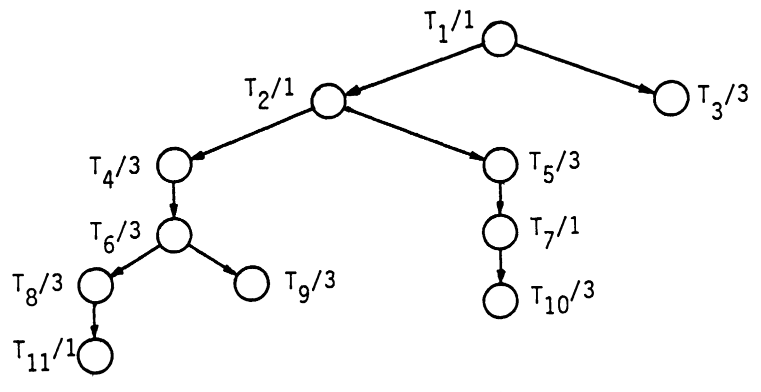


(a) The precedence tree $G$

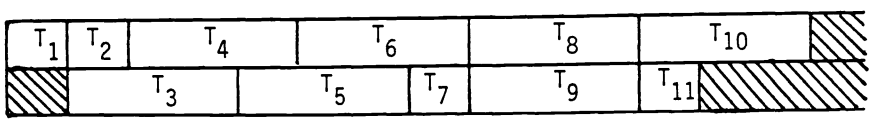


(b) The LSF schedule $S_1$ (also an optimal one)

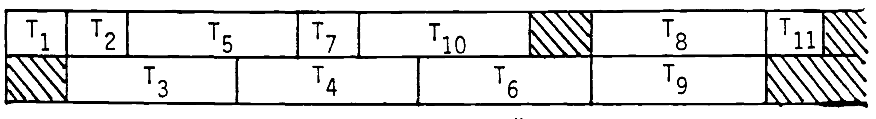
FIG. 5. *Case-3 degenerate* LSF *schedule* (*U cannot be delayed and* $D = 0$).



(a) The precedence tree $G$

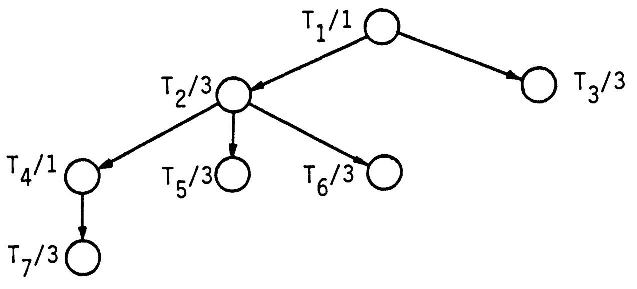


(b) The LSF schedule $S_1$

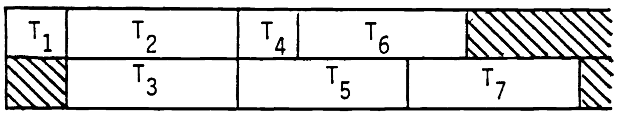


(c) The schedule $\ddot{S}_1$

FIG. 6. *Case-4 degenerate* LSF *schedule* (*U cannot be delayed and* $D > 0$. $\ddot{S}_1$ *has internal and external idle intervals*. $C(S_o^+(X), S_1^-(X))$ *is optimal*).



(a) The precedence tree $G$

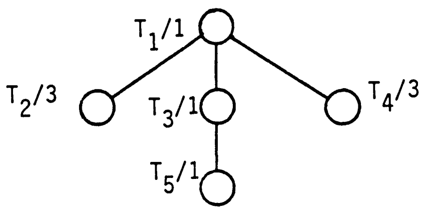


(b) The LSF schedule $S_1$

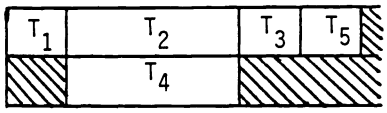
FIG. 7. *Case-5 degenerate* LSF *schedule.*

terms of this example, we simply start $T_2$ and $T_3$ at $\sigma - 5$, start $T_4$ when $T_3$ finishes, and start $T_5$ when $T_2$ finishes. The resulting schedule with length $\sigma - 1$ is shown in Fig. 8(c). We know by Lemma 3 that the improved schedule is optimal. This method can generally be used to improve any Case-6 degenerate schedules.

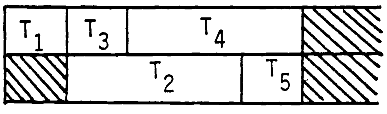
In the next three lemmas we show that the six cases defined above cover all possible degenerate schedules.



(a) The precedence tree $G$



(b) The LSF schedule $S_1$



(c) An optimal schedule

FIG. 8. *Case-6 degenerate* LSF *schedule.*

LEMMA 4. *If $S_l$ is degenerate and $L > 3$, then $S_l$ is either Case-1 or Case-2 degenerate.*

*Proof.* Since $S_l$ is degenerate, we have $L < \sigma$. Therefore, we can find a chain of $j+1$ tasks, $(T_{j+1}, T_j, \cdots, T_1)$, such that $f(T_1) = \sigma$, $\sigma - s(T_j) = L = \sum_{i=1}^{j} p(T_i)$ and $s(T_j) > f(T_{j+1})$. Consequently, $w(T_j) \geqq L$. Since $T_j$ is ready but not scheduled at $f(T_{j+1})$, no processor is idle in $[f(T_{j+1}), s(T_j)]$. Let $T'_1$ and $T'_2$ be the two tasks scheduled in $[s(T_j)-1, s(T_j)]$. Note that $T'_1$, $T'_2$, and $T_j$ are independent of one another. Since both $T'_1$ and $T'_2$ start earlier than $T_j$, at least one of $T'_1$ and $T'_2$ has weight no less than $T_j$. Without loss of generality, we assume $w(T'_1) \geqq w(T_j)$. We consider the following two cases separately: (a) $w(T'_1) \geqq w(T_j)$ and $w(T'_2) \geqq w(T_j)$; and (b) $w(T'_1) \geqq w(T_j)$ and $w(T'_2) < w(T_j)$.

Let $M$ $(M \geqq 0)$ denote the total execution time of the tasks that start after $f(T'_1)$, but are not successors of $T'_1$ or $T'_2$, and are not on the chain $(T_j, T_{j-1}, \cdots, T_1)$. The existence of the chain $(T_j, T_{j-1}, \cdots, T_1)$ implies that all successors of $T'_1$ and $T'_2$ and the tasks that are counted in $M$ should be executed in $[\text{MAX}, \sigma - N]$, where $\text{MAX} = \max\{f(T'_1), f(T'_2)\}$. That is,

$$(*) \qquad M + w(T'_1) - p(T'_1) + w(T'_2) - p(T'_2) = \sigma - N - \text{MAX}.$$

In the following we first show that case (a) is possible only if $L = 4$, $N = 2$, and $S_l$ is Case-2 degenerate. We then show that for case (b) it is impossible that $N = 3$, and when $N = 2$, $S_l$ is Case-1 degenerate.

For case (a), we let $M_1 = w(T'_1) - L$, $M_2 = w(T'_2) - L$, and $\text{MIN} = \min\{f(T'_1), f(T'_2)\}$. Note that $M_1 \geqq 0$ and $M_2 \geqq 0$. Since $f(T'_1) \geqq s(T_j)$ and $f(T'_2) \geqq s(T_j)$, we have $\sigma - \text{MIN} \leqq L$ and hence $\sigma - \text{MAX} \leqq L$. Substituting $M_1$ and $M_2$ into $(*)$, we obtain $M_1 + M_2 + N + M + L \leqq p(T'_1) + p(T'_2)$. Since $L > 3$ and $N \geqq 2$, we have $L + N \geqq 6$. Since $p(T'_1) + p(T'_2) \leqq 6$, we must have $L + N = 6$, or $L = 4$ and $N = 2$. It is also necessary that $M = M_1 = M_2 = 0$ and $p(T'_1) = p(T'_2) = 3$. That is, $w(T'_1) = w(T'_2) = w(T_j) = L = 4$. Consequently, $w(T'_1) - p(T'_1) = w(T'_2) - p(T'_2) = 1$ and $p(T_j) = 3$. Substituting the above values into $(*)$, we obtain $\sigma - \text{MAX} = 4$. However, $\sigma - \text{MAX} \leqq \sigma - \text{MIN} \leqq L = 4$. Therefore, $\text{MAX} = \text{MIN}$ and both $T'_1$ and $T'_2$ finish at $\sigma - 4$. In other words, $T'_1$ and $T'_2$ both start at $\sigma - 7$. From the above discussions, we see that $S_l$ is Case-2 degenerate.

For case (b), we show that it is impossible for $N = 3$. When $N = 2$, we show that the following five claims hold, from which we conclude that $S_l$ is Case-1 degenerate:

(1) $p(T'_1) = p(T'_2) = 3$.

(2) $f(T'_1) - s(T_j) = 1$ and $f(T'_2) = s(T_j)$.

(3) $w(T'_2) - p(T'_2) = 0$ and $M = 0$.

(4) $w(T'_1) = w(T_j) = h(T'_1) = h(T_j) = L$ and $p(T_j) = 3$.

(5) If $X$ is the immediate predecessor of $T'_1$, then $X$ is also the immediate predecessor of $T_j$. Moreover, $f(X) = s(T'_1)$ and $d(X) = f(X)$.

For case (b), we let $M_1 = w(T'_1) - L$. Since $w(T'_1) \geqq L$, we have $M_1 \geqq 0$. Since $\sigma = s(T_j) + L$ and since $\text{MAX} \geqq f(T'_1)$, we have from $(*)$ that

$$(**) \qquad M_1 + N + M + w(T'_2) - p(T'_2) + f(T'_1) - s(T_j) \leqq p(T'_1).$$

From $(**)$ and $N \geqq 2$, we see that $p(T'_1) \geqq 2$ and hence $p(T'_1) = 3$. Since $T'_1$, $T'_2$, and $T_j$ are independent of one another and since $w(T_j) > w(T'_2)$, it is necessary that $s(T'_1) > s(T'_2)$ and $p(T'_2) = 3$. Otherwise, $T_j$ would have been scheduled before $T'_2$ according to the first criterion of the LSF ordering. Therefore, $f(T'_2) = s(T_j)$ and $f(T'_1) > s(T_j)$. Since $p(T'_1) \leqq 3$, we must have $f(T'_1) - s(T_j) = 1$, $N = 2$, and $M = M_1 = w(T'_2) - p(T'_2) = 0$ for $(**)$ to hold. Substituting the above values into $(*)$, we obtain $w(T'_1) = L - 3 + p(T'_1)$. Since $w(T'_1) \geqq w(T_j) = L$, we have $p(T'_1) = 3$ and

$w(T'_1) = L$. Furthermore, we must have $h(T'_1) = h(T_j) = L$. Otherwise, $T_j$ would have been scheduled before $T'_1$ according to the third criterion of the LSF ordering. We have thus far proved claims (1)–(4).

We now prove claim (5). Let $X$ be the task executed in $[s(T'_2), s(T'_1)]$ in parallel with $T'_2$ on the other processor. Clearly, $f(X) = s(T'_1)$. Since $w(T'_1) \geqq 4$, $X$ cannot be a leaf task. That is, $X$ must be the immediate predecessor of some tasks scheduled after $f(X)$. There are, however, exactly two subtrees left at $f(X)$, namely the subtrees rooted at $T'_1$ and $T_j$. Since they are not executed at $s(T'_2)$ but they both have weight more than $T'_2$, neither can be independent with $X$. Therefore, they are both immediate successors of $X$. Thus, $w(suc(X)) = 2L \geqq 8$. By Lemma 1, we have $f(X) = d(X)$. □

LEMMA 5. *If $S_l$ is degenerate and $L = 3$, then $S_l$ is Case-3, Case-4, or Case-5 degenerate.*

*Proof.* Let $(T_j, \cdots, T_1)$ be the chain executing at the end of $S_l$, and $\hat{T}_1$ and $\hat{T}_2$ be the two tasks executing in $[s(T_j) - 1, s(T_j)]$. Since $L = 3$, $j$ must be 1 and $T_1$ must be a 3-unit leaf task. Otherwise, $T_j$ would be scheduled before $\hat{T}_1$ or $\hat{T}_2$. If $N = 3$, then both $\hat{T}_1$ and $\hat{T}_2$ finish at $\sigma - 3$. This implies that they are both leaf tasks. Furthermore, we must have $p(\hat{T}_1) = p(\hat{T}_2) = 3$; otherwise, $T_1$ would be scheduled before $\hat{T}_1$ or $\hat{T}_2$. This leads to $S_l$ being Case-3 degenerate.

If $N = 2$, then let $T'$ be the leaf task that finishes at $\sigma - 2$. There are two possibilities to consider: $p(T') = 1$ and $p(T') = 3$. If $p(T') = 1$, then both $\hat{T}_1$ and $\hat{T}_2$ finish at $\sigma - 3$. Since $L = 3$, $\hat{T}_1$ and $\hat{T}_2$ have to be independent with $T_1$ and at least one of them is a leaf task. Suppose $\hat{T}_1$ is a leaf task. We have $p(\hat{T}_1) = 3$; otherwise, $T_1$ would be scheduled before $\hat{T}_1$. Therefore, $S_l$ is Case-4 degenerate. If $p(T') = 3$, then there is only one task, say $\hat{T}_1$, that finishes at $\sigma - 3$. $\hat{T}_1$ must be a 3-unit leaf task. Hence, $S_l$ is Case-5 degenerate. □

LEMMA 6. *If $S_l$ is degenerate and $L = 2$, then $S_l$ is Case-6 degenerate.*

*Proof.* Since $L = 2$, the 2-unit-long chain executed at the end of $S_l$ is formed by two 1-unit tasks. Both processors must be executing tasks, say $T_1$ and $T_2$, in $[\sigma - 3, \sigma - 2]$ of $S_l$. $T_1$, $T_2$ and the task at the head of the 2-unit-long chain must be independent of one another. Since $L = 2$ and $N \geqq 2$, we have $N = 2$ in this case. Furthermore, $p(T_1) = p(T_2) = 3$. Otherwise, the task at the head of the 2-unit-long chain would be scheduled before either $T_1$ or $T_2$ by the LSF ordering. But this means that $S_l$ is Case-6 degenerate. □

Before we leave this section, we briefly review the operation of Algorithm ONE-THREE. Algorithm ONE-THREE first generates the LSF schedule $S_l$. If $S_l$ is not degenerate, then Algorithm ONE-THREE stops with $S_l$ being the optimal schedule. Otherwise, it tries to improve $S_l$ depending on which degenerate case $S_l$ is. For example, if $S_l$ is Case-6 degenerate, then it simply makes the rearrangement as described earlier. In the next section we give procedures to improve the other five degenerate cases.

## 4. Improving the degenerate schedules.

In this section we give procedures to improve the degenerate schedules. The procedure to improve Case-1 degenerate schedules is given in § 4.1. The procedure used to improve Case-2, Case-3, and Case-4 degenerate schedules is given in § 4.2, while the procedure to improve Case-5 degenerate schedules is given in § 4.3. As has been noted earlier, we do not need to consider Case-6 degenerate schedules since there is a simple way to improve such schedules. To simplify later proofs, in the following we give another characterization of degenerate schedules that enables us to make further assumptions concerning degenerate schedules.

Let $S_l$ be Case-$i$ degenerate, $1 \leqq i \leqq 5$, and let $[t_1, t_2]$ be the last internal idle interval of $S_l$ or the initial idle interval if $S_l$ does not have any internal idle interval. We locate two tasks in $S_l$, say $\tilde{X}$ and $\tilde{Y}$, by scanning $S_l$ backward from $\sigma - L$ to $t_2$. Let $\tilde{Y}$ be the first 1-unit task encountered in the scan such that $w(\tilde{Y}) \leqq 2$, and let $\tilde{X}$ be the dual task of $\tilde{Y}$ in $S_l$. If $\tilde{Y}$ does not exist, then we let $\tilde{X}$ be the task that finishes at $t_2$. We have the following lemma about $\tilde{X}$ and $\tilde{Y}$.

LEMMA 7. *The following three claims are true.* (7.1) $f(\tilde{X}) = f(\tilde{Y})$ *if* $\tilde{Y}$ *exists.* (7.2) *Every task in* $\Psi(\tilde{X})$ *is in suc* $(\tilde{X})$ *with only one possible exception—the* 1*-unit leaf task* $\hat{U}$ *executed in* $[\sigma - 3, \sigma - 2]$ *when* $S_l$ *is Case-*4 *degenerate.* (7.3) $f(\tilde{X}) = d(\tilde{X})$.

*Proof.* If $\tilde{Y}$ does not exist, then the lemma follows immediately from Lemma 3. Therefore, we assume that $\tilde{Y}$ exists. If (7.1) is not true, then $f(\tilde{X}) > f(\tilde{Y})$. Let $Z$ be the task executed in $[f(\tilde{Y}), f(\tilde{Y}) + 1]$ on the other processor concurrently with $\tilde{X}$. Then, we have $w(Z) \geqq 3$. For otherwise, $w(Z) \leqq 2$ and $p(Z) = 1$, and hence $Z$ would have been chosen instead of $\tilde{Y}$. Consequently, $Z$ and $\tilde{Y}$ must be independent of each other. Hence, $Z$ should have been scheduled before $\tilde{Y}$ in $S_l$ since $w(Z) > w(\tilde{Y})$. This is a contradiction. Therefore, we must have $f(\tilde{Y}) = f(\tilde{X})$.

We now consider the tasks, other than $\tilde{X}$ and $\tilde{Y}$, that can be ready at $s(\tilde{Y})$ in $S_l$. Let $\tilde{T}$ be such a task. Since $\tilde{T}$ is independent with $\tilde{X}$ and $\tilde{Y}$, we have $w(\tilde{T}) \leqq 2$; otherwise, $\tilde{T}$ would have been scheduled before $\tilde{Y}$. Consequently, $p(\tilde{T}) = 1$. This implies that $s(\tilde{T}) > \sigma - L$; otherwise, $\tilde{T}$ would have been chosen instead of $\tilde{Y}$. Thus, $\tilde{T}$ is scheduled in $[\sigma - L, \sigma]$. Since we are considering Case-$i$ degenerate schedules for $1 \leqq i \leqq 5$, the only time that this can happen is when $S_l$ is Case-4 degenerate and $\tilde{T} = \hat{U}$. Thus, (7.2) holds.

Finally, since (7.2) is true, we can apply Lemma 1 to $\tilde{X}$ to conclude that (7.3) holds, by noting that $w(suc\,(\tilde{X})) \geqq 9$ and $w(\tilde{Y}) \leqq 2$. □

From Lemma 7, we see that to improve a degenerate LSF schedule $S_l$, we cannot improve on the subschedule of $S_l$ before $f(\tilde{X})$. Thus, we only need to consider the subschedule of $S_l$ after $f(\tilde{X})$. To facilitate later proofs, we make the following assumptions: any degenerate LSF schedule mentioned hereafter has the following:

(A.1) No internal idle interval, and

(A.2) No 1-unit task with weight at most 2, except those scheduled in $[\sigma - L, \sigma]$.

Note that these two assumptions are made only to facilitate later proofs. Algorithm ONE-THREE does not need to locate $\tilde{X}$ and $\tilde{Y}$ in its operation.

**4.1. Case-1 degenerate.** In this section we consider how to improve Case-1 degenerate schedules. For the rest of this section we assume that $S_l$ is Case-1 degenerate. Recall that if $S_l$ is Case-1 degenerate, then there is a task $X$ with $f(X) = d(X)$. Furthermore, the successors of $X$ consist of two chains of tasks of length $L = \sigma - f(X) - 2$, headed by two 3-unit tasks *son* 1 and *son* 2. By assumptions (A.1) and (A.2), we see that *nsc* $(X)$ does not contain any 1-unit leaf tasks except possibly $X$. Moreover, $S_l$ does not contain any internal idle intervals. We show in the next lemma that $S_l$ is improvable if and only if $\sigma(S_o^+(X)) = d(X) + 1$.

LEMMA 8. *If* $\sigma(S_o^+(X)) = d(X) + 1$, *then the schedule* $C(S_o^+(X), S_l^-(X))$ *is one unit shorter than* $S_l$ *and hence it is optimal. Otherwise,* $S_l$ *is already optimal.*

*Proof.* If $\sigma(S_o^+(X)) = d(X) + 1$, then it is clear that $C(S_o^+(X), S_l^-(X))$ is one unit shorter than $S_l$. Conversely, if $\sigma(S_o^+(X)) \neq d(X) + 1$, then we must have $\sigma(S_o^+(X)) > d(X) + 1$. Suppose $S_l$ is not optimal. Let $S'$ be an optimal schedule such that $\sigma(S') = \sigma - 1$. Note that $S'$ does not contain any internal or external idle interval. Clearly, we cannot have $f(S', X) > d(X) + 1$; otherwise, $\sigma(S') > \sigma - 1$. Thus, we consider the following two possibilities: (a) $f(S', X) = d(X) + 1$; and (b) $f(S', X) = d(X)$.

For case (a), the tasks in *nsc* $(X)$ must finish by $f(S', X)$ in $S'$. Otherwise, we cannot finish all tasks in *suc* $(X)$ by $\sigma(S')$. But this means that there is a schedule for *nsc* $(X)$ with length $d(X) + 1$, contradicting the fact that $\sigma(S_o^+(X)) > d(X) + 1$. Thus, case (a) is impossible. For case (b), we let $Y'$ be the dual task of $X$ in $S'$. It is clear that $\Psi(S', X) = suc\,(X)$; otherwise, we cannot finish all tasks in *suc* $(X)$ by $\sigma(S')$. Thus, $Y'$ is a leaf task in *nsc* $(X)$ and hence $p(Y') = 3$. Since $w(nsc\,(X)) = 2(d(X) + 1)$, we have $f(S', Y') = f(S', X) + 2$. But then it is impossible to finish all tasks in *suc* $(X)$ by $\sigma(S')$, since *suc* $(X)$ consists of two chains of tasks each of which has length $L$. Thus, case (b) is also impossible. Hence, $S_l$ is already optimal. □

By Lemma 8, the procedure to improve a Case-1 degenerate schedule consists of recursively calling Algorithm ONE-THREE to construct an optimal schedule for the tasks in *nsc*$(X)$. If $\sigma(S_o^+(X)) = d(X) + 1$, then we output the schedule $C(S_o^+(X), S_l^-(X))$. Otherwise, we output $S_l$.

**4.2. Case-2, Case-3 and Case-4 degenerate.** In this section we consider how to improve Case-2, Case-3, and Case-4 degenerate schedules. For the rest of this section we assume that $S_l$ is Case-2, Case-3, or Case-4 degenerate. First, we need to define two pairs of tasks to facilitate our discussions. Recall the pair of tasks $X$ and $Y$ in a Case-1 degenerate schedule. $X$ has the properties that $w(\Psi(X) - suc\,(X)) = 0$ and $f(X) = d(X)$, and $Y$ is the dual task of $X$. For $S_l$ being Case-2, Case-3, or Case-4 degenerate, there is also a pair of tasks with properties similar to those of $X$ and $Y$. Abusing notation slightly, we also call them $X$ and $Y$. Let $K = \{T | f(T) \leqq \sigma - L$ and $w(\Psi(T) - suc\,(T)) \leqq 1\}$. Define $X$ to be the task in $K$ that finishes last in $S_l$ and $Y$ to be the dual task of $X$. Observe that $K$ cannot be empty since $\tilde{X}$ is in $K$. Consequently, $X$ is always defined and it can be located by scanning the schedule backward. However, $Y$ does not exist if $X$ finishes at the end of an idle interval. In this case, we let $Y$ be a dummy task with $f(Y) = f(X)$.

From the definition of $X$ we see that $\Psi(X) = suc\,(X)$, except in the case that $S_l$ is Case-4 degenerate and the 1-unit leaf task $\hat{U}$ executed in $[\sigma - 3, \sigma - 2]$ is not a successor of $X$. In this case we have $\Psi(X) = suc\,(X) \cup \{\hat{U}\}$. Furthermore, since $w(\Psi(X) - suc\,(X)) \leqq 1$ and since $w(suc\,(X)) \geqq 6$, it follows from Lemma 1 that $d(X) = f(X)$. From the definition of $Y$ we see that $w(suc\,(Y)) \leqq 1$ if $Y$ exists. Consequently, $Y$ is a leaf task if it exists, except in the case that $S_l$ is Case-4 degenerate and $\hat{U}$ is the son of $Y$. Let $D = f(Y) - f(X)$. Then $p(Y) = 3$ if $D > 0$.

The second pair of tasks, called $U$ and $V$, is defined as follows. Scan the interval $[f(X), \sigma - L]$ of $S_l$ backward, checking at time points $\sigma - 3i - L$ for $i \geqq 0$, until a 1-unit (nonleaf) task that finishes at the checking point is encountered or $f(X)$ is reached, whichever occurs first. If a 1-unit task is found, we denote it by $U$ and say that $U$ is defined. Otherwise, we say that $U$ is undefined. If $U$ is defined, we denote the dual task of $U$ in $S_l$ by $V$. It is easy to see that if $U$ is defined, then $f(U) = f(V)$ and $\Psi(U)$ consists of an odd number of 3-unit tasks plus three, zero, and one 1-unit task for $S_l$ being Case-2, Case-3, and Case-4 degenerate, respectively. Furthermore, if $U$ is defined and there is a 3-unit task in $\Psi(U)$ ready for execution at $s(U)$, then we say that $U$ can be delayed. Otherwise, we say that $U$ cannot be delayed. Note that $V$ could be the same as $Y$, but it cannot appear earlier than $Y$ in $S_l$.

We now consider how to improve $S_l$. We shall consider the following cases in the order given: (a) $U$ is undefined; (b) $U$ can be delayed; and (c) $U$ cannot be delayed. We prove in the following lemma that $S_l$ is optimal if $U$ is undefined. Figure 3 shows an example of this case.

LEMMA 9. *$S_l$ is optimal if $U$ is undefined.*

*Proof.* Since $U$ is undefined, there exists an integer $j$ such that $f(X) = f(Y) = \sigma - 3(j + 1) - L$, and for each $0 \leqq i \leqq j$, there are two 3-unit tasks in $\Psi(X)$ that finishes at

$\sigma - 3i - L$. This implies that $\Psi(X)$ consists of an odd number of 3-unit tasks plus three, zero, or one 1-unit tasks, depending on whether $S_l$ is Case-2, Case-3, or Case-4 degenerate. Hence, if $S_l$ is Case-3 or Case-4 degenerate, then it is optimal because: (a) $\Psi(X) = suc(X)$ or $\Psi(X) = suc(X) \cup \{\hat{U}\}$; and (b) $f(X) = d(X)$. If $S_l$ is Case-2 degenerate, then $\Psi(X) = suc(X)$. It is easy to see that $\hat{U}_1$, $\hat{U}_2$, and $\hat{U}_3$ executing in $[\sigma - L, \sigma]$ of $S_l$ are the only 1-unit tasks as well as the only leaf tasks in $suc(X)$. Thus, the only way to improve $S_l$ is to execute all three 1-unit leaf tasks on the same processor. However, it is easy to see that a longer schedule will result if they all execute on the same processor. Consequently, $S_l$ is optimal. □

Now, suppose $U$ can be delayed. We show in the following that $S_l$ can always be improved. We show that a new schedule $\dot{S}_l$ constructed from $S_l$ is one unit shorter than $S_l$. $\dot{S}_l$ is constructed as follows:

(1) Schedule all tasks except $U$ that finish by $f(U)$ in $S_l$ in the same manner as $S_l$.
(2) Schedule $U$ right after $V$ finishes on the same processor as $V$.
(3) Schedule the remaining tasks (the tasks in $\Psi(U)$) by the LSF rule. See Fig. 4 for an example of $\dot{S}_l$.

Let $\{U_1, U_2, \cdots, U_u\}(\{V_1, V_2, \cdots, V_v\})$ be the set of immediate successors of $U(V)$, and let $\{T_1, T_2, \cdots, T_t\}$ be the set of tasks ready for execution at $s(U)$ in $S_l$. Without loss of generality, we may assume that $T_t$ has the largest weight among all tasks in $\{T_1, T_2, \cdots, T_t\}$ and that it is chosen to start at $f(\dot{S}_l, V) - 1$ in $\dot{S}_l$. Since $T_t$ is ready for execution at $s(U)$ but it is not chosen, we must have $w(U) \geqq w(T_t)$. Moreover, we have $p(T_t) = 3$ and $f(\dot{S}_l, T_t) = f(\dot{S}_l, U) + 1$. We show in Lemma 10 that $\dot{S}_l$ is one unit shorter than $S_l$. To facilitate our proof, let us consider the tree $G_1$ shown in Fig. 9. Note that $V$ is treated as a 1-unit task in $G_1$, although it may be a 3-unit task in the original task system $G$. Let $TS_1$ be the set of tasks in $G_1$ and let $S_l(G_1)$ be the LSF schedule for $TS_1$ subject to the precedence constraints of $G_1$. Then, the schedule obtained by eliminating the root of $G_1$ from $S_l(G_1)$ is exactly the same as the partial schedule of $\dot{S}_l$ in the interval $[s(\dot{S}_l, V), \sigma(\dot{S}_l)]$. Thus, we can simply consider $S_l(G_1)$ instead of $\dot{S}_l$ when the discussion is restricted to the partial schedule of $\dot{S}_l$. In the proof of Lemma 10 we use them interchangeably.

LEMMA 10. *If $U$ can be delayed, then $\dot{S}_l$ is optimal.*

*Proof.* We first show that there is no internal idle interval in the partial schedule of $\dot{S}_l$ after $s(\dot{S}_l, V)$ (and hence in $S_l(G_1)$). Suppose not. Let $[t_1, t_2]$ be the first internal idle interval in the partial schedule of $\dot{S}_l$ in the interval $[s(\dot{S}_l, V), \sigma(\dot{S}_l)]$ and let $W$ be the task that finishes at $t_2$. The two tasks that start at $t_2$ cannot both be 1-unit tasks. Thus, $w(W) \geqq 4$ and $\Psi(\dot{S}_l, W) = suc(W)$. By Lemma 1, there is a chain of tasks executing continuously in the interval $[s(\dot{S}_l, V), t_2]$. This chain must be headed by $V$ in $G_1$; otherwise, we would have $w(T_t) > w(U)$. Let this chain be $(V, U, \hat{T}_j, \cdots, \hat{T}_1)$, where $\hat{T}_1 = W$. $\hat{T}_j$ must be an immediate successor of either $U$ or $V$ in the original task system $G$; otherwise, $w(T_j) > w(U)$ and $w(T_j) > w(V)$. Therefore, the chain $(T_j, \cdots, T_1)$ must execute continuously on one processor after $U$ in $S_l$ and the remaining tasks executed in the interval $[s(\dot{S}_l, V), t_2]$ in $S_l$, other than $U$ and $V$, must execute on the other processor in $S_l$. From the total execution time of the tasks executed in the interval, we see that $w(\Psi(W) - suc(W)) \leqq 1$. But this contradicts our definition of $X$ since $W$ should be chosen instead. Hence, it is impossible to have any internal idle interval in the partial schedule of $\dot{S}_l$ after $s(\dot{S}_l, V)$.

We now claim that if $\dot{S}_l$ has an external idle interval, then its length must be at most one unit. The claim follows from the fact that $\dot{S}_l$ does not have any internal idle interval after $s(\dot{S}_l, V)$ and the tasks that execute after $s(\dot{S}_l, V)$, other than $U$ and $V$, consist of an odd number of 3-unit tasks plus three, zero, or one 1-unit leaf tasks depending on

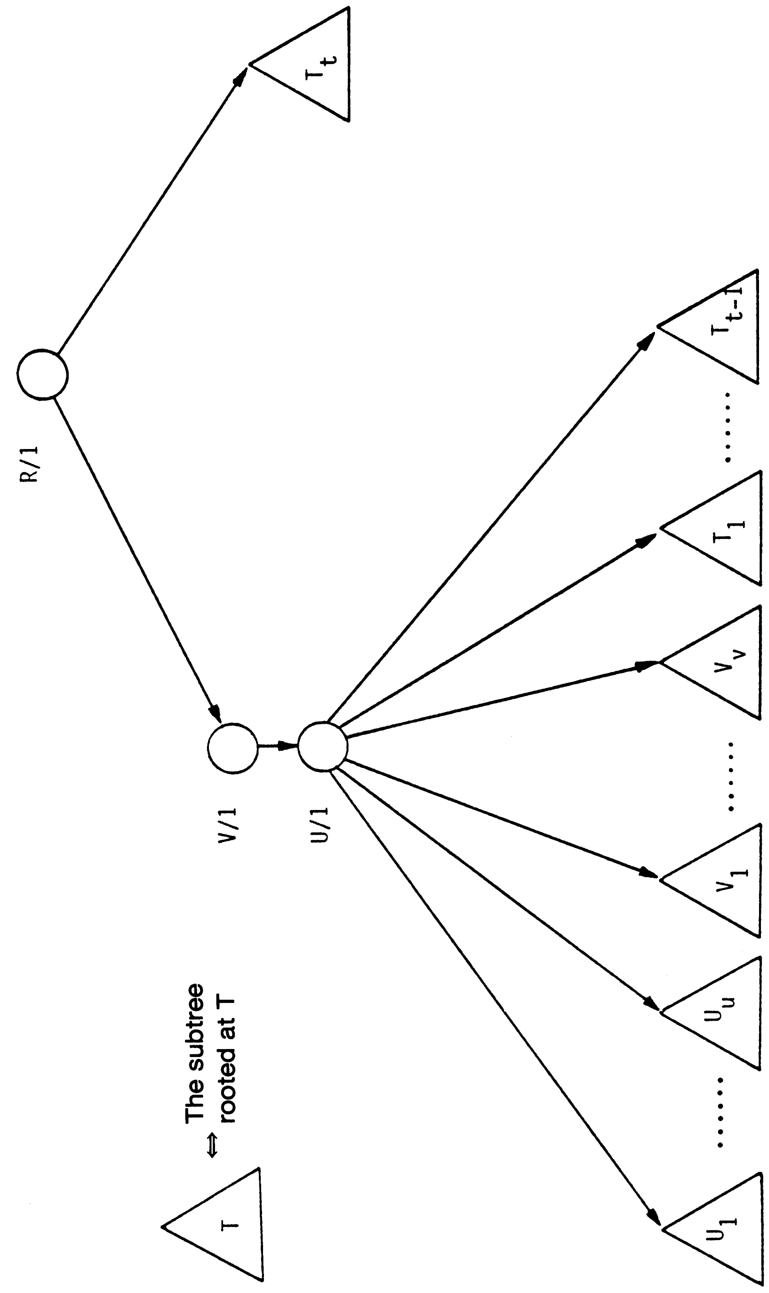


FIG. 9. *The precedence tree $G_1$.*

whether $S_l$ is Case-2, Case-3, or Case-4 degenerate. Since the length of the external idle interval of $\dot{S}_l$ is at most one unit, $\dot{S}_l$ must be one unit shorter than $S_l$. Hence, $\dot{S}_l$ is optimal. □

We now discuss the case when $U$ cannot be delayed. We show in Lemma 12 that $S_l$ is optimal if $D = 0$. An example of this situation is given in Fig. 5. If $D > 0$, then it can be shown that $X$ has exactly two immediate successors, excluding the possibility that $\hat{U}$ may also be an immediate successor of $X$ when $S_l$ is Case-4 degenerate. We denote the two immediate successors of $X$ by *son* 1 and *son* 2 and we assume that $w(son\ 1) \geqq w(son\ 2)$ and *son* 1 precedes *son* 2 in the LSF ordering of $TS$. Then, we have that $s(son\ 1) = f(X)$. Furthermore, since $|w(suc(U)) - w(suc(V))| \geqq 2$ and $U$ cannot be delayed, we can prove that $s(son\ 2) = f(Y)$, and that $d(son\ 1, UV_1) = f(UV_1) - s(son\ 1)$ and $d(son\ 2, UV_2) = f(UV_2) - s(son\ 2)$, where $\{UV_1, UV_2\} = \{U, V\}$. Note that $UV_1$ ($UV_2$) may be the same as *son* 1 (*son* 2) or it may be a successor of *son* 1 (*son* 2).

When $D > 0$, we construct another schedule $\ddot{S}_l$ from $S_l$ as follows:

(1) Schedule the tasks in $nsc(X)$ as in $S_l$.

(2) If $D = 1$ or $p(son\ 1) = 3$, then schedule *son* 2 right after $X$ finishes.

(3) If $D = 2$ and $p(son\ 1) = 1$, then schedule *son* 1 right after $X$ finishes and *son* 2 immediately following *son* 1.

(4) Schedule the remaining tasks in $suc(X)$ by the LSF rule.

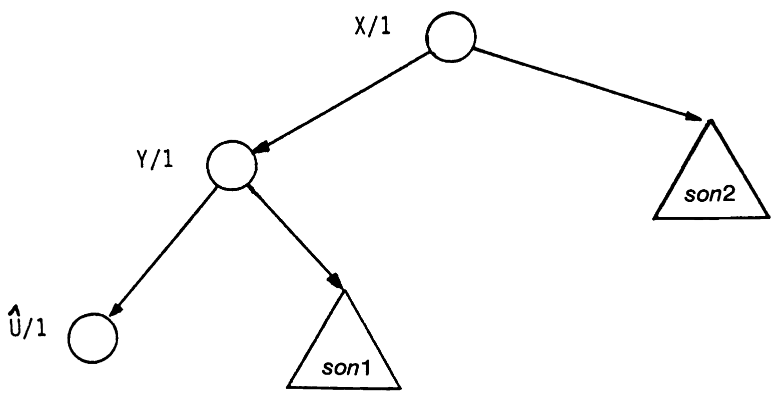
An example of $\ddot{S}_l$ can be found in Fig. 6. In Lemma 12 we show that if $\ddot{S}_l$ is not a better schedule than $S_l$, then $S_l$ must be Case-4 degenerate and $C(S_o^+(X), S_l^-(X))$ must be optimal. Note that if $D = 2$ and $p(son\ 1) = 1$, then *son* 1 has only one immediate successor, denoted by *son* 11, excluding the possibility that $\hat{U}$ may be another successor of *son* 1 when $S_l$ is Case-4 degenerate. Let $J = 2$ if $p(son\ 1) = p(son\ 2) = 3$ and $D = 2$, and $J = 1$ otherwise. Then, we have that $s(\ddot{S}_l, son\ 1) = s(son\ 1) + J$ if $D = 1$ or $p(son\ 1) = 3$, and $s(\ddot{S}_l, son\ 11) = s(son\ 11) + J$ otherwise.

To facilitate our proofs, let us consider the tree $G_2$ shown in Fig. 10. Define $TS_2$ to be the set of tasks in $G_2$ and $S_l(G_2)$ to be the LSF schedule of $TS_2$ subject to the precedence constraint of $G_2$. Then, the schedule obtained by deleting the root of $G_2$ from $S_l(G_2)$ is identical to the partial schedule of $\ddot{S}_l$ in the interval $[f(\ddot{S}_l, X), \sigma(\ddot{S}_l)]$. In the proofs of Lemmas 11 and 12, we refer to them interchangeably. We need the following lemma to prove Lemma 12.

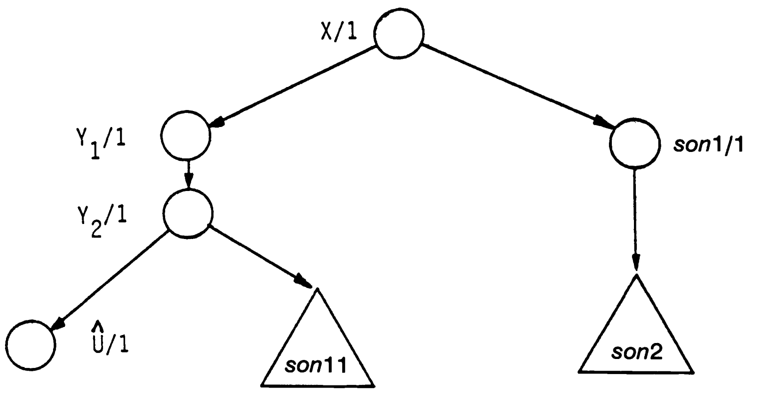
LEMMA 11. *Suppose that $U$ cannot be delayed and $D > 0$. Let $[t_1, t_2]$ be an internal or external idle interval and let $W$ be the task such that $f(\ddot{S}_l, W) = t_2$. Then, we have $d(W) = f(\ddot{S}_l, W) - J$, where $J$ is defined as above, under one of the following conditions*: (a) $[t_1, t_2]$ *is internal*; (b) $[t_1, t_2]$ *is external and* $t_2 - t_1 > 3$; *and* (c) $[t_1, t_2]$ *is external with $t_2 - t_1 = N$, where $N$ is the length of the external idle interval of $S_l$, and there is no internal idle interval in $\ddot{S}_l$.*

*Proof.* Let $Z$ be *son* 1 if $D = 1$ or $p(son\ 1) = 3$, and $Z$ be *son* 11 otherwise. Let $Z'$ be the immediate predecessor of $Z$. Then, it can be seen that $s(\ddot{S}_l, Z) = s(Z) + J$ and $d(Z') = f(Z') = s(Z)$. In the following we show that $W$ is a successor of $Z$ and $d(Z, W) = f(\ddot{S}_l, W) - s(\ddot{S}_l, Z)$. Therefore, $d(W) = d(Z') + d(Z, W) = f(\ddot{S}_l, W) + d(Z') - s(\ddot{S}_l, Z) = f(\ddot{S}_l, W) + f(Z') - s(Z) - J = f(\ddot{S}_l, W) - J$, and hence the lemma is proved.

Let $(T_{j+1}, T_j, \cdots, T_1)$ be the chain such that $f(\ddot{S}_l, T_{j+1}) < s(\ddot{S}_l, T_j)$, $f(\ddot{S}_l, T_{i+1}) = s(\ddot{S}_l, T_i)$ for $1 \leqq i \leqq j - 1$ and $T_1 = W$. If $s(\ddot{S}_l, T_j) \leqq f(\ddot{S}_l, Y)$, then one of the three tasks *son* 1, *son* 2, or $Y$ must be on the chain. Since $w(suc(Y)) \leqq 1$, $Y$ cannot be on the chain. If *son* 2 is on the chain, then we have $w(son\ 2) > w(son\ 1)$ under any one of the conditions (a), (b), or (c). This contradicts our assumption that



(a) $G_2$ when $D = 1$

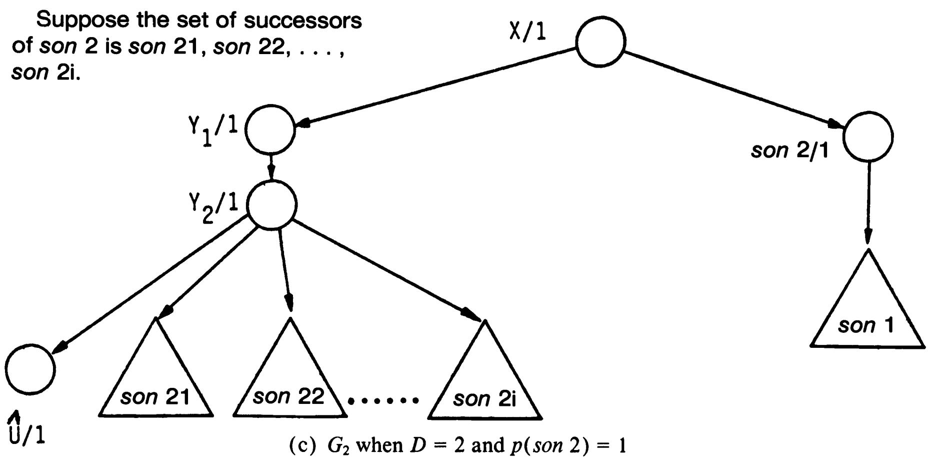


(b) $G_2$ when $D = 2$ and $p(son\ 1) = 1$

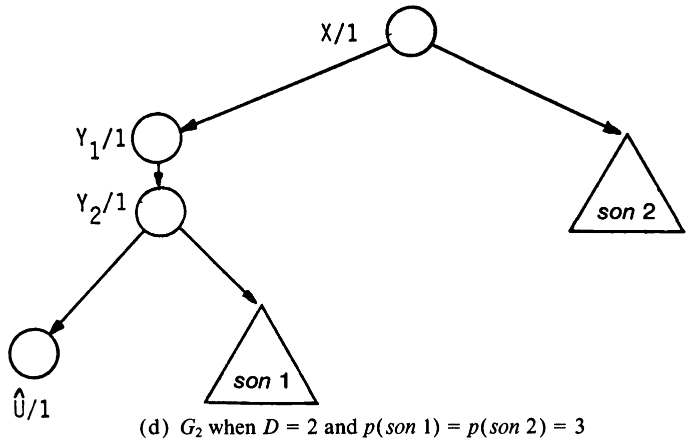
FIG. 10. *The precedence tree* $G_2$.

$w(son\ 2) \leqq w(son\ 1)$. Consequently, *son* 1 must be on the chain. From the construction of $\ddot{S}_l$, we can see that $T_j = son\ 1$ if $D = 1$ or $p(son\ 1) = 3$, and $T_j = son\ 11$ otherwise. Consequently, $W$ is a successor of $Z$ and $d(Z, W) = f(\ddot{S}_l, W) - s(\ddot{S}_l, Z)$.

To finish the proof, we show by contradiction that it is impossible to have $s(\ddot{S}_l, T_j) > f(\ddot{S}_l, Y)$. Suppose otherwise. Then, the chain $(T_{j+1}, T_j, \cdots, T_1)$ is also a chain in $G_2$. Now, let us consider $S_l(G_2)$. Under condition (a) or (b), this chain must execute continuously in $S_l(G_2)$ by Lemma 1. This contradicts our assumption that $f(\ddot{S}_l, T_{j+1}) < s(\ddot{S}_l, T_j)$. Under condition (c), if $S_l(G_2)$ is not degenerate, then there is a chain in $G_2$ with length as long as the length of $S_l(G_2)$, since the length of the external idle interval $[t_1, t_2]$ in $S_l(G_2)$ is $N$. This means that the chain $(T_{j+1}, T_j, \cdots, T_1)$ executes continuously in $S_l(G_2)$. This again contradicts our assumption. We now show that it is impossible to have $S_l(G_2)$ as degenerate. Since $|w(suc(U)) - w(suc(V))| \geqq 2$, we have $|f(\ddot{S}_l, U) - f(\ddot{S}_l, V)| = 2J$. Recall the following three facts: (1) $U$ is the 1-unit nonleaf task that finished last in $S_l$; (2) $\Psi(U)$ consists of an odd number of 3-unit tasks plus three, zero, or one 1-unit task, depending on whether $S_l$ is Case-2, Case-3, or Case-4 degenerate; and (3) $\ddot{S}_l$ has no internal idle interval. Since $|f(\ddot{S}_l, U) - f(\ddot{S}_l, V)| = 2J$ and from the above three facts, it is clear that $\ddot{S}_l$ cannot have the same end as $S_l$. Thus, $S_l(G_2)$ cannot be degenerate. Consequently, under condition (c), it is also impossible that $s(\ddot{S}_l, T_j) > f(\ddot{S}_l, Y)$. □


Suppose the set of successors of son 2 is son 21, son 22, . . . , son 2i.
X/1
$Y_1$/1
$Y_2$/1
son 2/1
son 1
son 21
son 22
son 2i
Û/1

(c) $G_2$ when $D = 2$ and $p(son\ 2) = 1$



(d) $G_2$ when $D = 2$ and $p(son\ 1) = p(son\ 2) = 3$

FIG. 10. (*Continued.*)

LEMMA 12. *Suppose that $U$ cannot be delayed. Then, $S_l$ is optimal if $D = 0$. If $D > 0$, then $\ddot{S}_l$ is one unit shorter than $S_l$ except when $S_l$ is Case-4 degenerate and $\ddot{S}_l$ has the same length as $S_l$. In this case, $C(S_o^+(X),\ S_l^-(X))$ is optimal.*

*Proof.* If $D = 0$, then the following can be shown: (1) $f(U) = d(U) = f(V) = d(V)$; (2) $\Psi(U)$ contains an odd number of 3-unit tasks; and (3) all tasks in $\Psi(U)$, except possibly $\hat{U}$ when $S_l$ is Case-4 degenerate, are in $suc\ (U) \cup suc\ (V)$. Hence, $S_l$ is optimal.

Now, consider that $D > 0$. If $\ddot{S}_l$ has no internal and no external idle interval, then $\ddot{S}_l$ must be one unit shorter than $S_l$. Thus, we may assume that $\ddot{S}_l$ has internal and/or external idle intervals. We consider the following two situations separately: $\ddot{S}_l$ has internal idle interval(s), and otherwise. We first prove that if $\ddot{S}_l$ has internal idle interval(s), then there is only one internal idle interval and its length is one unit. Furthermore, $S_l$ must be Case-3 or Case-4 degenerate. If $S_l$ is Case-3 degenerate, then we show that $\ddot{S}_l$ has no external idle interval. That is, $\ddot{S}_l$ is still 1 unit shorter than $S_l$. If $S_l$ is Case-4 degenerate, then we show that $J = 2$ (i.e., $D = 2$ and $p(son\ 1) = p(son\ 2) = 3$) and $C(S_o^+(X),\ S_l^-(X))$ is optimal. See Fig. 6 for an example. We then consider $\ddot{S}_l$ having external but no internal idle interval. In this case we show that $S_l$ must be Case-3 or Case-4 degenerate and that the length of the external idle interval is exactly one or two units, respectively. Therefore, if $S_l$ is Case-3 degenerate, then $\ddot{S}_l$ is still 1 unit shorter than $S_l$. If $S_l$ is Case-4 degenerate, then we show that $J = 2$, $d(\hat{U}) = \sigma - 2$, and $C(S_o^+(X), S_l^-(X))$ is optimal.

Suppose there are internal idle intervals in $\ddot{S}_l$. Let $[t_1, t_2]$ be the last one in $\ddot{S}_l$ and let $W$ be the task such that $f(\ddot{S}_l, W) = t_2$. Since $S_l$ has no internal idle interval, we have $t_2 > f(\ddot{S}_l, Y)$ and $W$ is in $suc(X)$. Furthermore, since $W$ has at least two successors one of which must be a 3-unit task, we have $w(suc(W)) \geqq 4$ and $f(W) < \sigma - 3$. Consequently, the interval $[f(X), f(W)]$ in $S_l$ contains no idle interval. Let $A = suc(X) - suc(W)$ and $q = 2(f(W) - f(X)) - D$. Then, we have $w(\Psi(W) - suc(W)) \leqq w(A) - q$. From the construction of $\ddot{S}_l$, we see that $w(A) = 2(f(\ddot{S}_l, W) - f(\ddot{S}_l, X)) - M - D$, where $M$ ($M \geqq 1$) is the total length of all internal idle intervals in $\ddot{S}_l$. Now, $q \geqq 2(d(W) - f(X)) - D$ (since $f(W) \geqq d(W)$) $= 2(f(\ddot{S}_l, W) - f(\ddot{S}_l, X)) - (2J + D)$ (by Lemma 12 and $f(X) = f(\ddot{S}_l, X)$). Consequently, $w(\Psi(W) - suc(W)) \leqq w(A) - q \leqq 2J - M \leqq 3$.

If $S_l$ is Case-2 degenerate, then it is easy to see that $w(\Psi(W) - suc(W)) \geqq 4$. Thus, it is impossible that $S_l$ is Case-2 degenerate. If $S_l$ is Case-3 or Case-4 degenerate, then $w(\Psi(W) - suc(W)) \geqq 3$. Therefore, $S_l$ must be Case-3 or Case-4 degenerate and $w(\Psi(W) - suc(W)) = 3$. Hence, we have $J = 2$ and $M = 1$. If $S_l$ is Case-3 degenerate, then it can be seen that $suc(W)$ consists of an even number of 3-unit tasks. Consequently, if $\ddot{S}_l$ has an external idle interval, then its length must be $6i$ for some $i \geqq 1$. However, by Lemma 11, this implies that there is a chain in the task system $G$ at least as long as $\sigma$. Therefore, by Observation 2, $\ddot{S}_l$ cannot have any external idle interval. If $S_l$ is Case-4 degenerate, then it can be seen that (1) the dual task of $W$ in $S_l$ finishes at the same time as $W$; (2) all tasks in $\Psi(W)$, except one 3-unit task, are in $suc(W)$; (3) $suc(W)$ contains $\hat{U}$ and an even number of 3-unit tasks; (4) $d(W) = f(W)$; (5) the length of the external idle interval of $\ddot{S}_l$ is one unit; and (6) the length of $S_l^-(X)$ is $\sigma - d(X) - 2$; i.e., $S_l^-(X)$ has no idle interval. Therefore, $\ddot{S}_l$ has the same length as $S_l$. Using a similar argument as in the proof of Lemma 8, it can be shown that $C(S_o^+(X), S_l^-(X))$ is optimal.

Finally, we consider the case when $\ddot{S}_l$ has an external but no internal idle interval. Let the length of its external idle interval be denoted by $N'$. Dictated by the total execution time of all tasks in $TS$, $N'$ must be equal to $N + 2i$ and $\sigma(\ddot{S}_l) = \sigma + i$ for some integer $i \geqq -1$. If $i \geqq 0$, then by Lemma 11, there is a chain in the task system $G$ no shorter than $\sigma + i - J \geqq \sigma - 2$. By Observations 1 and 2, this cannot hold for $S_l$ being Case-2 or Case-3 degenerate. Thus, we have $i = -1$ for these two cases; $N' = 0$ if $S_l$ is Case-2 degenerate, and $N' = 1$ if $S_l$ is Case-3 degenerate. Hence, $\ddot{S}_l$ is optimal for these two cases. If $S_l$ is Case-4 degenerate, then we must have $i = 0$ and $J = 2$ by Observation 3. In this case it can be seen that (1) $f(\ddot{S}_l, \hat{U}) = \sigma(\ddot{S}_l) = \sigma$; (2) $d(\hat{U}) = \sigma - 2$; and (3) the length of $S_l^-(X)$ is $\sigma - d(X) - 2$. Using the same argument as in the proof of Lemma 8, it can be shown that $C(S_o^+(X), S_l^-(X))$ is optimal. □

From Lemmas 9, 10, and 12, we see how to improve Case-2, Case-3, and Case-4 degenerate schedules. We first locate the task $X$ and then the task $U$ by scanning $S_l$ backward. If $U$ is undefined, then $S_l$ is already optimal by Lemma 9. Otherwise, if $U$ can be delayed, then we produce the schedule $\dot{S}_l$, which is optimal by Lemma 10. If $U$ cannot be delayed and $D = 0$, then $S_l$ is already optimal by Lemma 12. If $D > 0$, then we produce the schedule $\ddot{S}_l$. If $\ddot{S}_l$ is an improved schedule, then it is optimal. Otherwise, we produce the schedule $C(S_o^+(X), S_l^-(X))$, which is optimal by Lemma 12. Note that we need to recursively call Algorithm ONE-THREE to produce the schedule $S_o^+(X)$.

**4.3. Case-5 degenerate.** In this section we consider how to improve Case-5 degenerate schedules. For the remainder of this section we assume that $S_l$ is Case-5 degenerate.

Since $S_l$ is Case-5 degenerate, we see that there exists at least one 1-unit nonleaf task in $TS$ that is different from the root $R$. Therefore, we can apply the iterative pruning process described in [7]. That is, we iteratively prune the pair of 3-unit tasks farthest away from the root $R$ until a 1-unit task is first exposed as a leaf. We can prove that in the pruning process, we will never encounter the situation where the only leaf task left is a single 3-unit task. The proof is analogous to the proof of Lemma 4 in [7]. Let $G_3$ be the remainder of $G$ after the pruning process is completed and let $TS_3$ be the set of tasks in $G_3$. Let $S^-(G_3)$ denote the schedule for the pruned tasks in which the pair of tasks removed at the $j$th iteration of the pruning process is scheduled immediately after the pair removed at the $(j+1)$st iteration. In Lemma 13 we show that the schedule $C(S_o(G_3), S^-(G_3))$ is optimal for the tasks in $G$, where $S_o(G_3)$ is an optimal schedule for the tasks in $G_3$. Thus, the procedure to improve a Case-5 degenerate schedule consists of recursively calling Algorithm ONE-THREE to produce an optimal schedule $S_o(G_3)$ for the tasks in $G_3$ and then concatenate $S_o(G_3)$ with $S^-(G_3)$ to obtain an optimal schedule for the tasks in $G$. Let $S_l(G_3)$ denote the LSF schedule for the tasks in $G_3$.

LEMMA 13. *The schedule $C(S_o(G_3), S^-(G_3))$ is optimal for the tasks in $G$, where $S_o(G_3)$ and $S^-(G_3)$ are defined as above.*

*Proof.* By the same technique as in the proof of Lemma 5 in [7], we can show that $S_l(G_3)$ has no internal idle interval. Consequently, if $S_l(G_3)$ has an external idle interval, then its length must be a multiple of two. If $S_l(G_3)$ is not degenerate, then either $S_l(G_3)$ has no external idle interval or there exists a chain of length $\sigma(S_l(G_3))$ in $G_3$. In the former case, $S_o(G_3)$ has no internal or external idle interval. Hence, $C(S_o(G_3), S^-(G_3))$ is optimal. In the latter case, we can show that $C(S_o(G_3), S^-(G_3))$ is optimal by the same technique as in the proof of Lemma 5 in [7]. If $S_l(G_3)$ is degenerate, then $S_l(G_3)$ must be Case-1, Case-4, or Case-6 degenerate since it has no internal idle interval and the 1-unit task exposed as a leaf in the pruning process must finish last on some processor. Therefore, if $S_l(G_3)$ is improvable, then $S_o(G_3)$ has no internal or external idle interval and hence $C(S_o(G_3), S^-(G_3))$ is optimal. If $S_l(G_3)$ is not improvable, then it must be Case-1 or Case-4 degenerate. From the proofs of Lemmas 8 and 12, we see that there are only three situations in which Case-1 and Case-4 degenerate schedules cannot be improved. Combining the methods developed in the proofs of Lemma 5 in [7] and Lemma 8, we can show that $C(S_o(G_3), S^-(G_3))$ is also optimal in any of these three situations. □

**5. Conclusions.** In this paper we show that Algorithm ONE-THREE constructs an optimal 2-processor schedule for a set of $n$ tasks with a tree-structured precedence relation and execution time of 1 or 3 units. Algorithm ONE-THREE first constructs an LSF schedule. If the LSF schedule is not degenerate, then it is already optimal. Otherwise, Algorithm ONE-THREE attempts to improve it using the procedures given in § 4 until either an improvement is obtained or it reaches the conclusion that the original LSF schedule is already optimal. Algorithm ONE-THREE can be implemented to run in $O(n^2 \log n)$ time. This follows from the observations that a LSF schedule can be obtained in $O(n \log n)$ time and that Algorithm ONE-THREE needs to construct at most $n$ LSF schedules.

It is interesting to observe that Algorithm ONE-THREE can also be used to solve the case studied in [7]; the degenerate case described in [7] is actually Case-3 degenerate defined in this paper. We believe that the LSF algorithm defined in this paper can be quite useful in solving the case of execution time being 1 or $k$ units, where $k > 3$. The characterizations of LSF schedules given in § 2 are applicable to any $k$. For $k > 3$, we need to systematically classify the degenerate schedules and devise procedures to improve them. For future research efforts, it will be interesting to see how far this idea can go.

## REFERENCES

[1] E. G. COFFMAN, JR. AND R. L. GRAHAM, *Optimal sequencing for two processor systems*, Acta Inform., 1 (1972), pp. 200–213.

[2] J. DU AND J. Y-T. LEUNG, *Scheduling tree-structured tasks with restricted execution times*, Inform. Process. Lett., 28 (1988), pp. 183–188.

[3] M. FUJII, T. KASAMI, AND K. NINOMIYA, *Optimal sequencing of two equivalent processors*, SIAM J. Appl. Math., 17 (1969), pp. 784–789. Erratum, SIAM J. Appl. Math., 20 (1971), p. 141.

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

[5] T. C. HU, *Parallel sequencing and assembly line problems*, Oper. Res., 9 (1961), pp. 841–848.

[6] J. Y-T. LEUNG, *On scheduling independent tasks with restricted execution times*, Oper. Res., 30 (1982), pp. 163–171.

[7] K. NAKAJIMA, J. Y-T. LEUNG, AND S. L. HAKIMI, *Optimal two processor scheduling of tree precedence constrained tasks with two execution times*, Performance Evaluation, 1 (1981), pp. 320–330.

[8] J. D. ULLMAN, NP-*complete scheduling problems*, J. Comput. System Sci., 10 (1975), pp. 384–393.

[9] ———, *Complexity of sequencing problems*, in Computer and Job-Shop Scheduling Theory, E. G. Coffman, Jr., ed., John Wiley, New York, 1976, pp. 139–164.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 197-218, May 1989

# ON FACTORABLE EXTENSIONS AND SUBGRAPHS OF PRIME GRAPHS*

JOAN FEIGENBAUM† AND RAMSEY W. HADDAD‡

**Abstract.** Cartesian-factorable extensions and subgraphs of prime graphs are investigated. It is shown that minimal factorable extensions and maximal factorable subgraphs are not unique and that finding them is NP-hard even, in the case of minimal factorable extensions, if the prime graph in question is required to be a tree. Tight bounds on the density of a prime graph's minimal factorable extension are derived. A dynamic programming algorithm is given for finding factorable extensions of certain types of trees.

**Key words.** product graphs, NP-completeness, dynamic programming

**AMS(MOS) subject classifications.** 68R10, 68Q15, 05C70

## 1. Introduction.

The *cartesian product* $G = G_1 \square G_2$ of undirected graphs $G_1$ and $G_2$ is an undirected graph with node set $V(G) = V(G_1) \times V(G_2)$ and edge set $E(G) = \{(x_1, y_1) — (x_2, y_2): (x_1 = x_2 \text{ and } y_1 — y_2 \in E(G_2)) \text{ or } (y_1 = y_2 \text{ and } x_1 — x_2 \in E(G_1))\}$. More concretely, form the product $G_1 \square G_2$ by substituting a copy of $G_2$ for each node in $G_1$ and drawing in the trivial isomorphism between each pair of copies corresponding to adjacent nodes. Figure 1 shows the cartesian product of a three-node chain and a triangle.

Cartesian multiplication is commutative (i.e., $G_1 \square G_2 \cong G_2 \square G_1$) and associative (i.e., $(G_1 \square G_2) \square G_3 \cong G_1 \square (G_2 \square G_3)$). Every connected graph has a set of irreducible factors that are unique up to order [9]; hence, irreducible graphs are called *primes*. Graphs that are not prime are called *factorable*. A simple counting argument shows that almost all graphs are prime [4]; more precisely, the fraction of labeled graphs on $n$ nodes that are factorable tends to $2^{-cn^2}$, for some positive constant $c$, as $n$ tends to infinity. The prime factors of a finite, connected graph can be found in polynomial time [5], [11].

Here, we consider the following question. Given a prime graph $G$, find a graph $H$ such that we have the following:

(a) $G$ is a subgraph of $H$ ($G \subseteq H$);
(b) $H$ is factorable;
(c) Among all graphs satisfying (a) and (b), $H$ has the fewest nodes; and
(d) Among all graphs satisfying (a), (b), and (c), $H$ has the fewest edges.

Such an $H$ is called a *minimal factorable extension* of $G$. Similarly, we define a *maximal factorable subgraph* $H$ of $G$ as a graph $H \subseteq G$ such that $H$ is factorable, has at least as many nodes as any factorable subgraph of $G$, and has at least as many edges as any factorable subgraph with that many nodes. Figure 2 shows a prime graph $G$, a minimal factorable extension $H$, and a maximal factorable subgraph $J$.

These questions arise in the design of computer networks and multiprocessing machines. It is easy to construct factorable graphs that have low diameter and a large number

* Received by the editors July 24, 1987; accepted for publication (in revised form) October 18, 1988. Some of the results in this paper were presented at the SIAM Discrete Mathematics Conference, Clemson, South Carolina, May 1986.

† AT&T Bell Laboratories, Murray Hill, New Jersey 07974. Part of the work of this author was done at Stanford University, Stanford, California 94305 and was supported by a Xerox Corporation Fellowship and a grant from the AT&T Bell Laboratories Graduate Research Program for Women.

‡ Stanford University, Stanford, California 94305. Part of the work of this author was done at AT&T Bell Laboratories; the rest was supported by National Science Foundation grants DCR-83-08109 and CCR-86-10181.

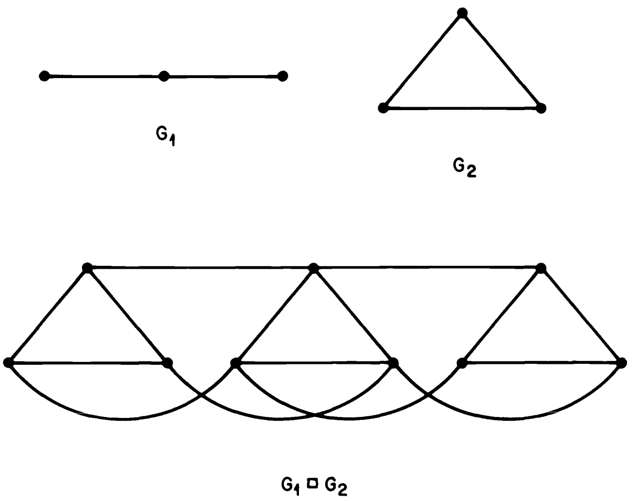


FIG. 1. *The product of a three-node chain and a triangle.*

of alternative paths between any pair of nodes; both of these characteristics are desirable in networks, the first for fast communication and the second for fault tolerance. Of course, real computer networks are not always built according to a systematic design, but often take shape gradually, in response to ad hoc demands for service. In order to improve communication rates or fault tolerance in a network represented by an arbitrary graph $G$, it may suffice to find a factorable graph $H$ that is "close" to $G$ (e.g., a small factorable extension or a large factorable subgraph), determine an optimal strategy for $H$, and then modify it to obtain a nearly optimal strategy for $G$. See, e.g., [2] for an application of these ideas to the construction of (static) routing tables.

Multiprocessors, on the other hand, *are* built according to systematic designs. Therefore, a great deal of effort goes into designing multiprocessor architectures that are sufficiently flexible to allow common programs that were originally written for a unipro-

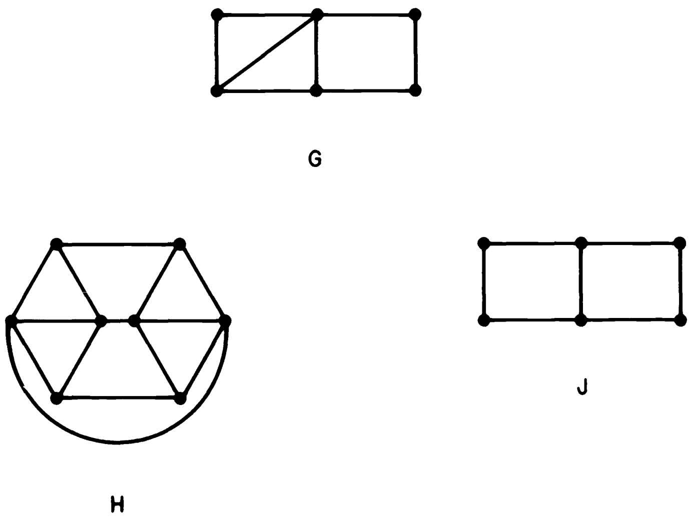


FIG. 2. *A prime graph $G$, its maximal factorable subgraph $J$, and its minimal factorable extension $H$.*

cessor to be decomposed and run on the multiprocessor in a way that exploits locality in the original program. If the original program is represented by a graph $G$, then the nodes of $G$ can be thought of as local computations (i.e., computations that can be run on a single processor) and the edges of $G$ can be thought of as requirements for communication between these local computations. If the structure of the multiprocessor is represented by a graph $H$, then the assignment of local computations in the original program to individual processors in the multiprocessor is modeled by a homomorphism of $G$ into $H$. Special kinds of product graphs, e.g., hypercubes and their generalizations, have achieved wide acceptance as multiprocessor architectures. Therefore, considerable attention has been paid to the question of finding mappings (if possible, homomorphisms) from (usually prime) graphs that represent programs into product graphs that represent multiprocessor architectures; see [10] for a thorough discussion of this issue. Our problem of finding smallest factorable extensions of arbitrary graphs is a further abstraction of this line of research.

We show that minimal factorable extensions and maximal factorable subgraphs are not unique and that finding them is NP-hard, even, in the case of minimal factorable extensions, if the prime graph in question is required to be a tree. We derive tight bounds on the density of $G$'s minimal factorable extension as a function of the density of $G$. We show how to use dynamic programming to find a factorable extension $H$ with $|V(H)|$ as small as possible in the special case in which $G$ is a tree and $H$ has one factor of size two or three. Finally, we show that any binary tree $T$ on more than four nodes has a minimal factorable extension $H \cong K_2 \square H'$, where $|V(H)| = 2 \cdot \lceil |V(T)|/2 \rceil$.

**2. Nonuniqueness.** The existence of a minimal factorable extension for every prime graph $G$ is self-evident:

$$G \subset \overline{K_2} \square G, \tag{1}$$

where $E_n = \overline{K_n}$ is the graph with $n$ nodes and no edges. In fact, (1) gives us the upper bound

$$|V(H)| \leqq 2|V(G)|,$$

which is tight, because the complete graph $K_n$ has no factorable extension smaller than $\overline{K_2} \square K_n$.

In general, a prime graph may have more than one minimal factorable extension and more than one maximal factorable subgraph. To see this, we construct infinite families of examples of nonuniqueness.

For each prime integer $p \geqq 7$, there is a prime graph $G_p$ with at least two nonisomorphic minimal factorable extensions $H'_p$ and $H''_p$. Suppose that $p = 2k - 1$. Let $V(G_p) = \{1, 2, \cdots, p\}$ and $E(G_p) = \{i — (i+1)\colon 1 \leqq i \leqq k-1 \text{ or } k+1 \leqq i \leqq p-1\} \cup \{i — (i+k)\colon 1 \leqq i \leqq k-2\}$. To obtain $V(H'_p)$ or $V(H''_p)$ from $V(G_p)$, just add one new node labeled $2k$. The graph $H'_p$ is just the product of $K_2$ and a path on $k$ nodes; the edge set $E(H'_p)$ is $E(G_p) \cup \{(k-1) — (2k-1), k — 2k, (2k-1) — 2k\}$. The graph $H''_p$ is the product of $K_2$ and a tree consisting of a path on $k-2$ nodes and two leaves both adjacent to the $(k-2)$nd node in the path; the edge set $E(H''_p)$ is $E(G_p) \cup \{(k-2) — 2k, k — (2k-2), (2k-1) — 2k\}$. Obviously, $H'_p$ and $H''_p$ are factorable and nonisomorphic. To see that they are minimal, observe that any factorable extension of $G_p$ has to be connected, and that the sparsest connected factorable graphs on $p+1$ nodes have $p - 1 + (p+1)/2$ edges, as do both $H'_p$ and $H''_p$. (Any such graph is of the form $K_2 \square T$, where $T$ is a tree on $(p+1)/2$ nodes.) Figure 3 shows $G_7$, $H'_7$, and $H''_7$.

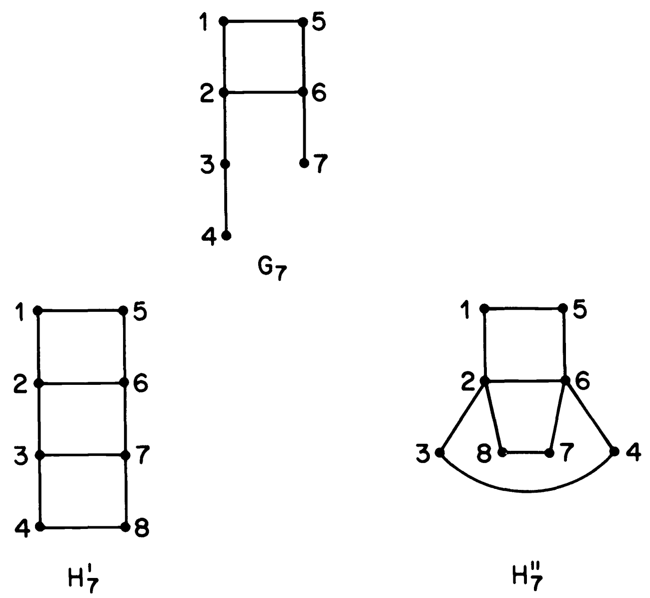


FIG. 3. *A graph $G_7$ and two nonisomorphic minimal factorable extensions $H'_7$ and $H''_7$.*

Prime graphs may also have two or more nonisomorphic maximal factorable subgraphs. For each $k \geqq 4$, let $J_{2k}$ be the prime graph on $2k$ nodes $\{1, 2, \cdots, 2k\}$ with edge set

$$\{i - (i+1)\colon 1 \leqq i \leqq k-1 \text{ or } k+1 \leqq i \leqq 2k-1\}$$
$$\cup \{i - (i+k)\colon 1 \leqq i \leqq k\} \cup \{1 - (k+3), 3 - (k+1)\}.$$

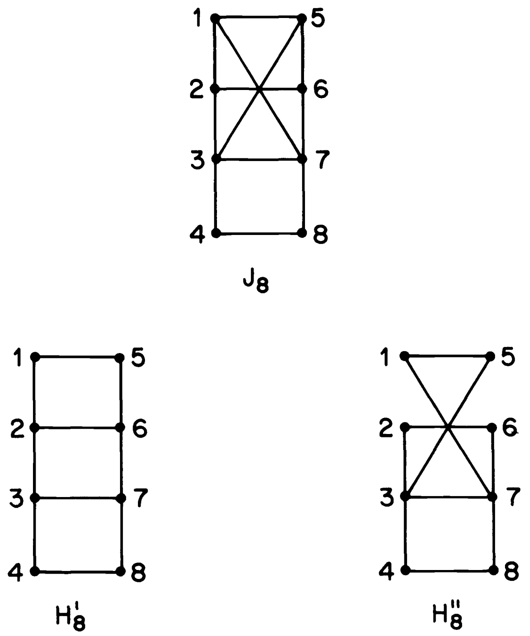


FIG. 4. *A graph $G_8$ and two nonisomorphic maximal factorable subgraphs $H'_8$ and $H''_8$.*

(The primality of $J_{2k}$ can be demonstrated quite straightforwardly using the factoring algorithm in [4]; we leave this proof to the reader, because the machinery used in the algorithm is considerable and not necessary for the rest of this paper.) Two nonisomorphic maximal factorable subgraphs are $H'_{2k}$ and $H''_{2k}$, which have the same structure as the factorable extensions described above. Specifically $V(H'_{2k}) = V(H''_{2k}) = V(J_{2k})$,

$$E(H'_{2k}) = E(J_{2k})\backslash\{1 — (k+3), 3 — (k+1)\},$$

and $E(H''_{2k}) = E(J_{2k})\backslash\{1 — 2, (k+1) — (k+2)\}$. Figure 4 shows $J_8$, $H'_8$, and $H''_8$.

## 3. NP-completeness.

In this section, we prove that the computational problems of finding smallest factorable extensions and largest factorable subgraphs are NP-hard. In yes/no form, the first problem is the following:

SMALLEST FACTORABLE EXTENSION (SFE).
*Input*: A graph $G$ and positive integers $N$ and $E$.
*Question*: Is there a factorable graph $H$ such that $G \subseteq H$, $|V(H)| \leqq N$, and $|E(H)| \leqq E$?

Suppose that $|V(H)| = N$ and $H = H_1 \square H_2$, where $|V(H_1)| = k$. We can view the process of extending $G$ to $H$ as follows. First, add $N - |V(G)|$ isolated nodes to $G$ and call this enlarged node set $V(H) = \{x_1, \cdots, x_N\}$. Next, find an optimal, legal partition of $V(H)$ into $k$ pieces, each of size $N/k$, and number the vertices $(i, j)$, $1 \leqq i \leqq k$, $1 \leqq j \leqq N/k$. Finally, add edges within the pieces so that they form isomorphic copies of the right factor $H_2$, and add edges between pieces so that their connection pattern is given by the left factor $H_1$.

The partition that corresponds to the numbering of $V(H)$ by ordered pairs $x_1 = (i_1, j_1), \cdots, x_N = (i_k, j_{N/k})$, is legal if

$$(i,j) — (i',j') \in E(G) \Rightarrow i = i' \text{ or } j = j',$$

and it is optimal if it minimizes

$$(2)\qquad |E(H)| = \frac{N}{k}\cdot\left|\bigcup_j \{\{i,i'\} : (i,j) — (i',j) \in E(G)\}\right| + k\cdot\left|\bigcup_i \{\{j,j'\} : (i,j'\} : (i,j) — \in E(G)\}\right|.$$

The first condition says that the edges inherited from $G$ are a *subset* of the edges of a cartesian-product graph. The set $\{\{i, i'\} : (i, j) — (i', j) \in E(G)\}$ represents $H_1$-edges that go between copies $i$ and $i'$ of $H_2$, whereas the set $\{\{j, j'\} : (i, j) — (i, j') \in E(G)\}$ represents the contribution of the $i$th piece in the partition to the edge set $E(H_2)$.

If $x_1 = (i_1, j_1), \cdots, x_N = (i_k, j_{N/k})$ is an optimal, legal partition of $V(H)$ into $k$ pieces of size $N/k$, then $x_1 = (j_1, i_1), \cdots, x_N = (j_{N/k}, i_k)$ is an optimal, legal partition into $N/k$ pieces of size $k$—this is to be expected, because the first partition yields the extension of $G$ to $H_1 \square H_2$, where $|V(H_1)| = k$, and the second yields the extension to $H_2 \square H_1$. These graphs are isomorphic; hence either both extensions are minimal or neither is. Therefore, we assume for the rest of this section that

$$(3)\qquad k \leqq \sqrt{N}.$$

The NP-completeness proof for the SFE problem proceeds in two stages. First we prove the NP-completeness of the *parameterized* SFE problem, in which we ask about the existence of an extension with factors of specified sizes. We then reduce this specific version of the problem to the general version.

PARAMETERIZED SMALLEST FACTORABLE EXTENSION (PSFE).
*Input*: A graph $G$ and positive integers $N$, $E$, and $k$.
*Question*: Is there a graph $H$ such that $G \subseteq H$, $H \cong H_1 \square H_2$, $|V(H_1)| = k$, $|V(H)| \leqq N$, and $|E(H)| \leqq E$?

LEMMA 3.1. *The* PSFE *problem is* NP-*complete, even for the values* $N = |V(G)|$, $E = (N/2)^2$, *and* $k = 2$.

*Proof.* Note first that the problem is certainly in NP: given a graph $H$, and a one-to-one mapping of $G$ into $H$, we can verify in polynomial time that $H$ has the required properties and that $G \subseteq H$.

For the special case in which $N = |V(G)|$ and $E = (N/2)^2$, we are asking whether $G$ can be extended to *any* graph with a 2-node factor without adding new nodes. The parameter $E = (N/2)^2$ imposes no restriction, because it is the maximum number of edges that an $N$-node cartesian-product graph with a 2-node factor can have; the maximum is achieved by $K_2 \square K_{N/2}$. Thus all we ask is whether $G$ admits a legal partition into two pieces of size $N/2$—i.e., can we divide $V(G)$ into equal-sized sets $S$ and $T$ such that, for any $s \in S$, there is *at most one* $t \in T$ such that $s - t \in E(G)$ and vice versa? If there is such a partition, then the sets $S$ and $T$ can be filled out to the copies of $H_2$ and the edges connecting $S$ to $T$ can be viewed as part of a one-to-one correspondence between the copies.

The NP-hardness part of the proof is by reduction from one-in-three 3SAT with no negated literals [6]. Given an instance ($C_1 = x_{11} \vee x_{12} \vee x_{13}$, $C_2 = x_{21} \vee x_{22} \vee x_{23}, \cdots$, $C_n = x_{n1} \vee x_{n2} \vee x_{n3}$) of this problem, we construct an instance of PSFE as follows: $V(G)$ contains one node $x_{ij}$ for each literal and three nodes, say $C_i$, $C_i^1$, and $C_i^2$, for each clause. For each pair $i \neq j$, draw in the edge $C_i - C_j$, and for each $i$, draw in $C_i - x_{ij}$, for $j \in \{1, 2, 3\}$, and $x_{ij} - C_i^j$ for $j \in \{1, 2\}$. In addition, if literals $x_{ij}$ and $x_{i'j'}$ are the same variable, draw in edge $x_{ij} - x_{i'j'}$. (See Fig. 5.)

Suppose there is a satisfying assignment that gives the value TRUE to the literals $x_{1j_1}, x_{2j_2}, \cdots, x_{nj_n}$; all the other literals must be assigned FALSE, by definition of one-in-three 3SAT. Then we get a legal partition of $V(G)$ by putting the TRUE literals $x_{ij_i}$ and the "padding nodes" $C_i^j$ in one set and the FALSE literals and "clause nodes" $C_i$ in the other.

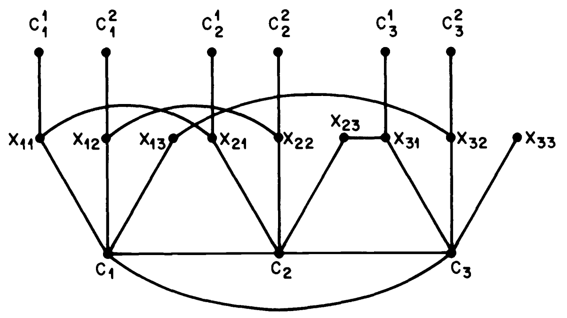


FIG. 5. *Instance of* PSFE *corresponding to the one-in-three* 3SAT *instance* ($C_1 = x \vee y \vee z$, $C_2 = x \vee y \vee w$, $C_3 = w \vee z \vee v$), *as in the proof of Lemma* 3.1.

Conversely, suppose that $S \sqcup T$ is a legal partition of $V(G)$. Any clique in $G$ of size three or more must lie completely in $S$ or completely in $T$. The clause nodes $C_1, \cdots, C_n$ form a clique; hence we can assume without loss of generality that they are all in $S$. For each $i$, at most one of the neighbors $x_{i1}$, $x_{i2}$, and $x_{i3}$ of $C_i$ can be in $T$. Because $S$ has the $n$ members $C_1, \cdots, C_n$, the only way it can have size $3n = |V(G)|/2$ is if, for each $i$, exactly two of the $x_{ij}$'s are in $S$ and the other is in $T$. Consider the assignment that gives the value TRUE to all of the literals in $T$ and FALSE to all of those in $S$. We have just argued that there is exactly one TRUE literal in each clause. Might the assignment be inconsistent—in other words, if literals $x_{ij}$ and $x_{i'j'}$ represent the same variable, could one of the corresponding nodes, say $x_{ij}$, be in $T$ and the other in $S$? This could not happen with a legal partition $S \sqcup T$, because $x_{ij} \in T$ would have two neighbors in $S$, namely $C_i$ and $x_{i'j'}$. Hence, each legal partition corresponds to a satisfying one-in-three assignment. $\square$

Our original SFE problem, which is also obviously in NP, asks for less information than the parameterized version, and hence it is conceivable that the unparameterized version could be easier. The following argument, however, shows that this is not the case.

We give a simple method of reducing an instance $(G, N = |V(G)|, E = (N/2)^2, k = 2)$ of PSFE to an instance $(G', N', E')$ of SFE. Let $H = K_2 \square K_{N+1}$ and $G'$ be the disjoint union $G \sqcup H$. Let $N' = N + |V(H)| = 3N + 2$ and $E' = E + |E(H)| = (N/2)^2 + N^2 + 2N$.

If $(G, N, E, 2)$ is a yes-instance of PSFE, then $G \subset K_2 \square J$, where $|V(J)| = N/2$. This implies that $G' \subset K_2 \square (J \sqcup K_{N+1})$; therefore, $(G', 3N + 2, E')$ is a yes-instance of SFE.

Now suppose that $(G', 3N + 2, E')$ is a yes-instance of SFE. Suppose that $H' = H'_1 \square H'_2$ is a factorable extension of $G'$ that witnesses this fact. Consider the nodes of one of the large cliques $K_{N+1}$ in $G'$. As in the proof of Lemma 3.1, either all of these nodes lie within one copy of $H'_2$ or each of them lies in a different copy. In either case, the fact that $|V(H')| = |V(G')| = 3N + 2$ implies that one of $|V(H'_1)|$ and $|V(H'_2)|$ is two, because the other is at least $N + 1$; say $|V(H'_1)| = 2$. Since each of the copies of $K_{N+1}$ must lie in a different copy of $H'_2$, the only way to complete a partition of $V(H')$ into two copies of $H'_2$ is to put $N/2$ nodes of $G$ into each copy, which can only be done if $(G, N, E, 2)$ is a yes-instance of PSFE.

This completes the proof of the following theorem.

THEOREM 3.1. *The* SFE *problem is* NP-*complete*.

Wagner has obtained several related results in which restrictions are placed on the classes of product graphs considered [10]. Recall that the *k-dimensional hypercube* is the cartesian product of $k$ graphs, each of which is just $K_2$; Wagner uses the term *k-dimensional B-cube* to mean the cartesian product of $k$ graphs, each of which is isomorphic to $B$. He proves that the following problems about extending prime graphs to product graphs are also NP-complete. Given a graph $G$ and an integer $k$, is $G$ a subgraph of the $k$-dimensional hypercube? Given a graph $G$, is it a subgraph of any hypercube (see also [3])? Given a graph $G$ and an integer $k$, is there a graph $B$ such that $G$ is a subgraph of the $k$-dimensional $B$-cube?

We turn now to the problem of finding large factorable subgraphs. The yes/no form of the problem is the following.

LARGEST FACTORABLE SUBGRAPH (LFS).
*Input*: A graph $G$ and positive integers $N$ and $E$.
*Question*: Is there a factorable graph $H$ such that $H \subseteq G$, $|V(H)| \geqq N$, and $|E(H)| \geqq E$?

THEOREM 3.2. *The* LFS *problem is* NP-*complete.*

*Proof.* LFS is obviously in NP. To prove completeness, we reduce an instance $(G, k)$ of CLIQUE [6] to an instance $(G', N, E)$ of LFS.

Let $n$ equal $|V(G)|$. Construct graphs $A$, $B$, and $C$, where $A \cong K_k$, $B \cong \overline{K_n}$, and $C \cong \overline{K_k}$. Next construct a graph $D$, where $V(D)$ is $V(A) \cup V(B) \cup V(C) \cup V(G)$, and $E(D)$ consists of all edges internal to the subgraphs $A$, $B$, $C$, and $G$, plus all edges $x — y$ such that $x \in V(A)$ and $y \in V(G)$, all edges $x — y$ such that $x \in V(B)$ and $y \in V(C)$, and $n$ edges that present a bijection between $V(G)$ and $V(B)$. The total number of edges in $D$ is therefore

$$|E(D)| = |E(G)| + \binom{k}{2} + 2kn + n.$$

Let $b$ be the smallest integer greater than $|E(D)|$ such that $n + k + b$ is prime; we know by a theorem of Chebyshev [8] that the size of $b$ is polynomial in $n$, and thus we can find $b$ in polynomial time by trial division of successive odd integers. Let $F$ be a graph isomorphic to the product $K_2 \square K_b$. The graph $G'$ in our target instance of LFS is the disjoint union of $D$ and $F$. The parameter $N$ is $|V(G')| = 2(n + k + b)$, and the parameter $E$ is

$$2\binom{b}{2} + b + 2\binom{k}{2} + k + 2\left\lceil \frac{k}{2} \right\rceil \left\lfloor \frac{k}{2} \right\rfloor + n.$$

(See Fig. 6.)

This construction of $G'$ is rather complicated, but it can be analyzed easily. The goal is to show that any factorable subgraph of $G'$ that is large enough must induce a $k$-clique that is completely contained in $G$.

First assume that $(G, k)$ is a yes-instance of CLIQUE. We will construct a factorable subgraph $H$ of $G'$ to show that $(G', N, E)$ is a yes-instance of LFS. $H$ will have the form $K_2 \square H'$; to do this, we have to partition $V(G')$ into two sets $S$ and $T$ that form the vertex sets of the copies of the right factor $H'$ and show that $E(G')$ contains edges that give a one-to-one correspondence between $S$ and $T$.

Start by putting $V(G)$ and the nodes of one of the $b$-cliques of $F$ into $S$ and putting $V(A)$ and the other $b$-clique of $F$ into $T$. The two $b$-cliques will correspond to each other in $H$, and they can be ignored during the rest of the construction of $S$ and $T$. Let the

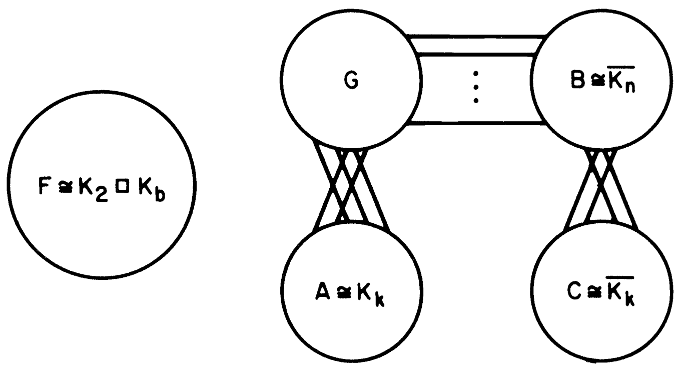
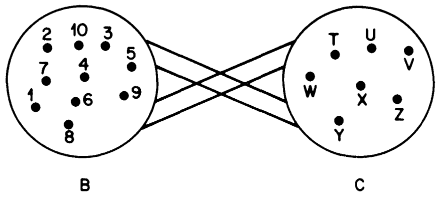


FIG. 6. *Instance of* LFS *corresponding to the* CLIQUE *instance* $(G, k)$, *as in the proof of Theorem* 3.2. *All possible edges are present between subgraphs A and G and between subgraphs B and C. Each node in G is adjacent to a unique node in B and vice versa.*

nodes of $A$ correspond to those of a $k$-clique $X$ in $G$; we know that such an $X$ exists, because $(G, k)$ is a yes-instance of CLIQUE, and that $E(G')$ contains a bijection between $V(A)$ and $V(X)$, because it contains *all* possible edges between $V(A)$ and $V(G)$. Each of the $n - k$ nodes in $V(G)\backslash V(X)$ is adjacent to exactly one node in $V(B)$. Put those $n - k$ nodes of $V(B)$ into $T$. The remaining $k$ nodes of $V(B)$ and the $k$ nodes of $V(C)$ induce a subgraph of $G'$ that contains $K_2 \square K_{\lceil k/2 \rceil, \lfloor k/2 \rfloor}$. There may be more than one way to identify the two copies of the complete bipartite subgraph $K_{\lceil k/2 \rceil, \lfloor k/2 \rfloor}$, but they all have the property that one copy contains $\lceil k/2 \rceil$ nodes of $B$ and $\lfloor k/2 \rfloor$ nodes of $C$ and the other contains $\lceil k/2 \rceil$ nodes of $C$ and $\lfloor k/2 \rfloor$ nodes of $B$ (see Fig. 7). We can choose any partition into two copies of $K_{\lceil k/2 \rceil, \lfloor k/2 \rfloor}$ and put the nodes of one copy into $S$ and the nodes of the other copy into $T$. $E(G')$ will contain edges that give a bijection between the two copies, because it contains all possible edges between $B$ and $C$.

It is clear that we have identified a factorable subgraph $H$ of $G'$ that has $|V(G')|$ nodes and has the form $K_2 \square H'$. The only additional fact to check is that $E(H)$ is large enough. The subgraph $F$ of $G'$ is completely contained in $H$, and it contributes $2\binom{b}{2} + b$ edges. The subgraph $A$, the clique $X$ of $G$, and the edges between them contribute $2\binom{k}{2} + k$ edges. There are $n - k$ edges between $G\backslash X$ and $B$, and the subgraph isomorphic to $K_2 \square K_{\lceil k/2 \rceil, \lfloor k/2 \rfloor}$ contributes $2\lceil k/2 \rceil \lfloor k/2 \rfloor + k$ edges. Summing these contributions, we get that $E(H)$ is exactly equal to the parameter $E$ of our LFS instance. This completes the proof that if $(G, k)$ is a yes-instance of CLIQUE, then $(G', N, E)$ is a yes-instance of LFS.

Now suppose that $(G', N, E)$ is a yes-instance of LFS. The factorable subgraph $H$ is such that $|V(H)| = |V(G')| = N$. Because $N = 2(n + k + b)$, and $n + k + b$ is prime, $H$ must be of the form $H'' \square H'$, where $|V(H'')| = 2$. Our first goal is to show that $H'' \cong K_2$.



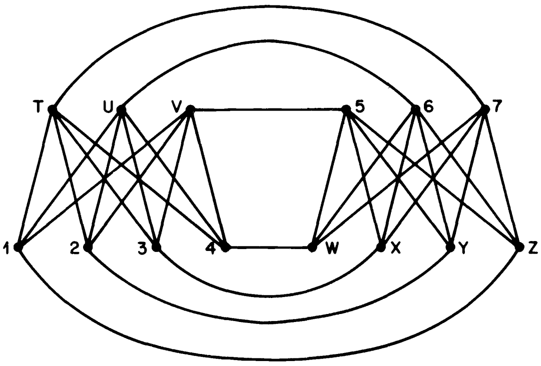


FIG. 7. *Subgraph, isomorphic to* $K_2 \square K_{\lfloor k/2 \rfloor, \lceil k/2 \rceil}$, *of the part of the instance induced by B and C. Here* $k = |V(C)| = 7$. *All such subgraphs must have one copy of* $K_{\lfloor k/2 \rfloor, \lceil k/2 \rceil}$ *that has* $\lceil k/2 \rceil$ *nodes of B and* $\lfloor k/2 \rfloor$ *of C and another copy that has* $\lfloor k/2 \rfloor$ *nodes of B and* $\lceil k/2 \rceil$ *of C, because there are no edges internal to B or C.*

$H''$ is either $K_2$ or $\overline{K_2}$. If $H'' \cong \overline{K_2}$, then the partitions of $V(G')$ into two sets $S$ and $T$ that maximize the size of $|E(H)|$ all have one of the large cliques $K_b$ in $S$ and the other in $T$. The total number of edges we can achieve in this case is at most

$$2\binom{b}{2} + |E(D)| < 2\binom{b}{2} + b < E.$$

This is not enough to make $(G', N, E)$ a yes-instance of LFS. Hence, the left factor of the subgraph $H$ is $K_2$.

The proof that $(G, k)$ is a yes-instance of CLIQUE can be completed by showing that $|E(H)|$ will be less than $E$ unless the subgraph $G$ contributes a clique of size $k$ to one of the copies $S$ and $T$ of the right factor $H'$. $E(H)$ has to contain a bijection between $S$ and $T$. None of the nodes of $F$ can correspond to a node outside of $F$, because there are no edges between $F$ and $G' \backslash F$. So $F$ contains a factorable subgraph of $H'$. This subgraph can contribute at most $2\binom{b}{2} + b$ edges to $E(H)$.

Each of the nodes in $C$ must correspond to a node in $B$. The factorable subgraph of $H$ made up of $k$ nodes from $C$ and $k$ from $B$ can contribute at most $2\lceil k/2 \rceil \lfloor k/2 \rfloor +$ $k$ edges to $E(H)$. This maximum contribution is achieved by the graph in Fig. 7.

The $n - k$ nodes of $B$ that do not correspond to nodes in $C$ must correspond to nodes in $G$. Because each node in $B$ is adjacent to exactly one node in $G$ and there are no edges internal to $B$, no assignment of these $2(n - k)$ nodes to $S$ and $T$ can produce a factorable subgraph that contributes more than $n - k$ edges, the bijection edges, to $E(H)$.

The remaining $k$ nodes of $G$ must correspond to nodes in $A \cong K_k$. So far, we have at most $m = 2\binom{b}{2} + b + 2\lceil k/2 \rceil \lfloor k/2 \rfloor + k + (n - k)$ edges in $E(H)$. The difference between $E$, the third parameter of the LFS instance, and $m$ is $2\binom{k}{2} + k$. We can only make up this deficit if the entire clique $A$ is contained in $H$ and the corresponding subgraph of $G$ is a $k$-clique. □

**4. Density.** The NP-completeness results of the previous section show that it is computationally difficult to find, for an arbitrary prime graph $G$, the *exact* size of its minimal factorable extension $H$. In this section, we derive bounds on $|E(H)|$ in terms of $|E(G)|$—that is, how much "filling in" is really needed to extend $G$ to a factorable graph?

Let $n = |V(G)|$, $e = |E(G)|$, $N = |V(H)|$, and $E = |E(H)|$. If $H = H_1 \square H_2$, where $|V(H_1)| = k$, then (2) yields an upper bound on $E$, namely,

$$\begin{aligned} E &= \frac{N}{k} \cdot \left| \bigcup_j \{\{i, i'\} : (i,j) - (i',j) \in E(G)\} \right| \\ &\quad + k \cdot \left| \bigcup_i \{\{j, j'\} : (i,j) - (i,j') \in E(G)\} \right| \\ &\leqq \frac{N}{k} \cdot \left| \bigcup_j \{\{i, i'\} : (i,j) - (i',j) \in E(G)\} \right| \\ &\quad + k \cdot \sum_{i=1}^{k} |\{\{j, j'\} : (i,j) - (i,j') \in E(G)\}| \\ &\leqq \frac{N}{k} \cdot \binom{k}{2} + k \cdot |E(G)| \leqq Nk + ke. \end{aligned}$$

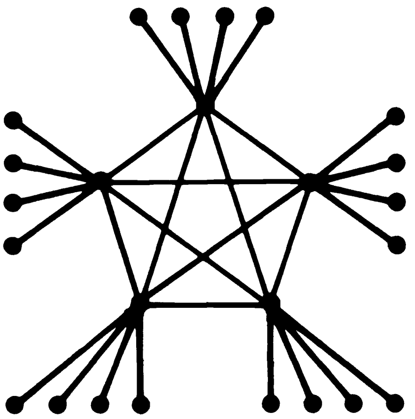


FIG. 8. *Prime graph requiring maximum possible fill-in.*

First consider the case in which $G$ has only linearly many edges. By (3), we can take $k \leqq \sqrt{N} \leqq \sqrt{2n}$. Thus we have the upper bound

$$e \leqq c_1 n \Rightarrow E \leqq c_2 n^{3/2},$$

where $c_1$ and $c_2$ are constants.

To show that this bound is tight up to constant factors, we construct an infinite family of prime graphs $G_n$ with $|E(G_n)| = O(n)$ whose minimal factorable extensions $H_n$ have $|E(H_n)| = \Omega(n^{3/2})$. Let $n = p^2$, where $p$ is prime, and take $V(G_n) = \{(i,j): 1 \leqq i, j \leqq p\}$. Let $E(G_n) = \{(1,j) — (1,j'): j \neq j'\} \cup \{(1,j) — (i,j): i \neq 1\}$. Thus $e = |E(G_n)| = \binom{p}{2} + p(p-1) < 2p^2 = O(n)$. Figure 8 shows $G_{25}$.

By drawing in edges $(i,j) — (i,j'), j \neq j', i > 1$, we can fill out $G_n$ to the product $K_p \square K_{1,p-1}$ of the $p$-clique and the $p$-node star (see Fig. 9). This does not require adding any new nodes; so $N = n = p^2$. The number of edges in $K_p \square K_{1,p-1}$ is $p(p-1) + \binom{p}{2}p = \Omega(p^3) = \Omega(n^{3/2})$.

Is the partition of $V(G_n)$ given by the original node-numbering optimal? Because $N = p^2$, all legal partitions must consist of $p$ pieces each of size $p$. The nodes $(1, 1), \cdots, (1, p)$ of $G_n$ form a clique; hence, in any legal partition, either all of these nodes are in

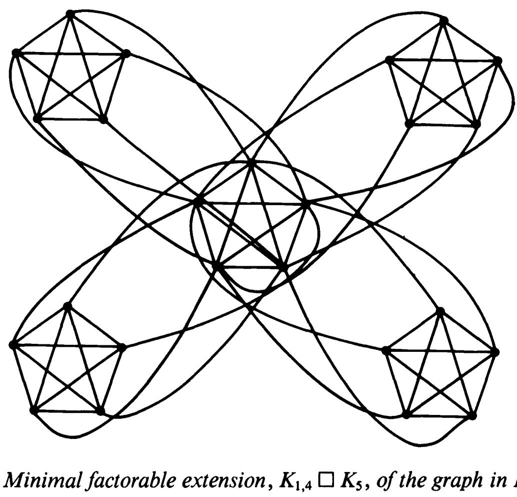


FIG. 9. *Minimal factorable extension*, $K_{1,4} \square K_5$, *of the graph in Fig.* 8.

the same piece or they are each in a different piece. If they are all in the same piece, then the right factor of the corresponding factorable extension is $K_p$—this means that the $p$ copies of the right factor contribute $p\binom{p}{2} = \Omega(n^{3/2})$ edges to $E(H_n)$. On the other hand, if each of the nodes $(1, 1), \cdots, (1, p)$ is in a different piece of the partition, then the *left* factor of the corresponding extension is $K_p$; once again, this gives a lower bound of $p\binom{p}{2}$ on $|E(H_n)|$.

These results can be generalized to graphs with $e \leqq c_1(n^{2-1/m})$, where $c_1$ is a constant and $m$ is any positive integer. Let $G$ be such a graph, and let $H = H_1 \square H_2$ be a minimal factorable extension of $G$, where $k = |V(H_1)|$. Recall that $k \leqq \sqrt{N} \leqq \sqrt{2n}$. Let $E_1$ be the number of edges of $H$ that go between copies of $H_2$, and let $E_2$ be the number of edges internal to copies of $H_2$. First observe that

$$E_1 \leqq \binom{k}{2}\frac{N}{k} < kN \leqq (2n)^{3/2}.$$

Next observe that

$$E_2 = k \cdot |E(H_2)| \leqq k\binom{N/k}{2} < \frac{N^2}{k}.$$

If $k \geqq N^{1/2m}$, then $E_2 \leqq N^2/N^{1/2m} = N^{2-1/2m}$, and so

$$E = E_1 + E_2 \leqq (2n)^{3/2} + (2n)^{2-1/2m} \leqq c_2' n^{2-1/2m}. \tag{4}$$

Now suppose that $k < N^{1/2m}$. As in the linear case, $E_2 \leqq ke$. So we have immediately that $E_2 \leqq N^{1/2m} \cdot c_1 n^{2-1/m} \leqq c_1(2n)^{2-1/2m}$. Thus

$$E = E_1 + E_2 \leqq (2n)^{3/2} + c_1(2n)^{2-1/2m} \leqq c_2'' n^{2-1/2m}. \tag{5}$$

Together (4) and (5) give the desired upper bound $e \leqq c_1 n^{2-1/m} \Rightarrow E \leqq c_2 n^{2-1/2m}$.

To show that this bound is tight, we generalize the family $G_n$ given in the proof of the linear case. This will give us prime graphs with $e = O(n^{2-1/m})$ whose minimal factorable extensions have $E = \Omega(n^{2-1/2m})$. Let $n = p^{2m}$, where $p$ is prime, and let $V(G_n) = \{(i, j): 1 \leqq i \leqq p,\ 1 \leqq j \leqq p^{2m-1}\}$. Once again, let $E(G_n) = \{(1, j) — (1, j'): j \neq j'\} \cup \{(1, j) — (i, j): i \neq 1\}$. So

$$e = \binom{p^{2m-1}}{2} + p^{2m-1}(p-1) = O(p^{4m-2}) = O(n^{2-1/m}).$$

An argument completely analogous to the one for the linear case shows that the minimal factorable extension of $G_n$ is $K_{1,p-1} \square K_{p^{2m-1}}$, which has $E = \Omega(n^{2-1/2m})$.

The preceding discussion gives us the following theorem.

THEOREM 4.1. *For all integers $m \geqq 1$, if $G$ is a prime graph with minimal factorable extension $H$, $|V(G)| = n$ and $|E(G)| = c_1(n^{2-1/m})$, for some constant $c_1$, then $|E(H)| \leqq c_2 n^{2-1/2m}$, for some other constant $c_2$. There is an infinite class of graphs $G_n$, with $|V(G_n)| = n$ and $|E(G_n)| = O(n^{2-1/m})$, for which $|E(H_n)| = \Omega(n^{2-1/2m})$.*

## 5. Factorable extensions of trees.

In this section, we consider a simplified version of the PSFE problem and obtain a polynomial-time dynamic-programming solution in two special cases.

PARAMETERIZED FACTORABLE EXTENSIONS OF TREES (PFET).
*Input*: A tree $T$ and positive integers $k$ and $m$.
*Question*: Is there a factorable graph $H \cong H_1 \square H_2$, such that $T \subset H$, $|V(H_1)| = k$, and $|V(H)| = m$?

Note that the analogous special case of the LFS problem (i.e., the case in which the input graph is required to be a tree) does not make sense, because it is impossible for a tree, which is acyclic, to have a factorable subgraph.

The PFET problem generalizes the following question considered in [1] and [10]: given a tree, what is the dimension of the smallest hypercube in which it can be embedded? Wagner has shown that the following decision problem is NP-complete [10]: given a tree $T$ and an integer $k$, is $T$ a subgraph of the $k$-dimensional hypercube? His proof uses the fact that the 3-partition problem is NP-complete, even if the input is expressed in unary [6].

3-PARTITION.

*Input*: A set $A = \{a_1, \cdots, a_{3n}\}$ of $3n$ distinct elements and an integer weight $s(a)$ for each $a \in A$, satisfying the conditions that $\sum_{a \in A} s(a) = nB$ and $B/4 < s(a) < B/2$, for all $a \in A$.

*Question*: Is there a partition of $A$ into disjoint sets $\{a_{i0}, a_{i1}, a_{i2}\}$, $1 \leqq i \leqq n$, such that $s(a_{i0}) + s(a_{i1}) + s(a_{i2}) = B$, for each $i$?

We also use 3-partition to prove the following.

THEOREM 5.1. *The* PFET *problem is* NP-*complete*.

*Proof*. The PFET problem is obviously in NP. We exhibit a many-to-one reduction from 3-partition to PFET. Let $(A = \{a_1, \cdots, a_{3n}\}, s, B)$ be an instance of 3-partition, expressed in unary. For notational convenience, let $j_1 = j_2 = 3n$. To construct the tree $T$ in the corresponding instance of PFET, we start by constructing $3n + 1$ disjoint stars. Let $S_0$ be a star with $Bj_1 + 3j_2 + (n + 1)$ nodes and, for $1 \leqq i \leqq 3n$, let $S_i$ be a star with $s(a_i)j_1 + j_2 + (n + 1)$ nodes. The vertex set of $T$ is just the disjoint union of the vertex sets $V(S_i)$ and the edge set of $T$ is the disjoint union of the edge sets $E(S_i)$, together with an edge from the root of $S_0$ to one leaf of each of the $S_i$, $i > 0$. (See Fig. 10.)

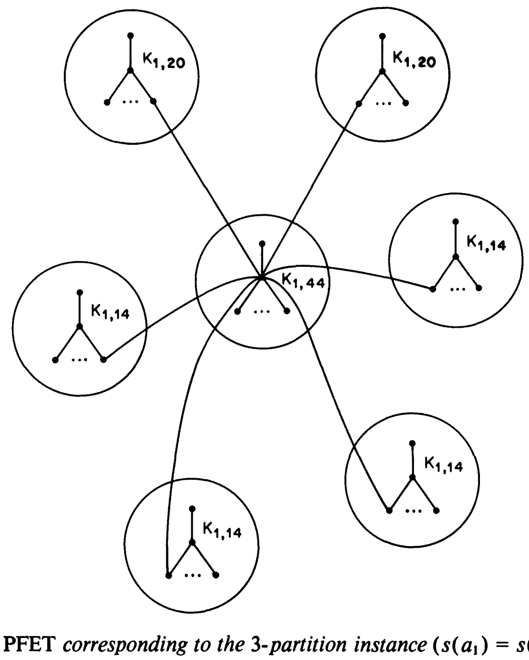


FIG. 10. *Instance of* PFET *corresponding to the* 3-*partition instance* $(s(a_1) = s(a_2) = s(a_3) = s(a_4) = 1$, $s(a_5) = s(a_6) = 2$, $B = 4)$, *as in the proof of Theorem* 5.1.

To finish the specification of the PFET instance, let $k = n + 1$ and $m = |V(T)| = (Bj_1 + 3j_2 + 3n + 1)\cdot(n + 1)$. It should now be clear why it is important that the numbers in the 3-partition instance be expressed in unary; otherwise, this reduction would entail an exponential blow-up in the size of the instance.

It is easy to see that, if $(A, s, B)$ is a yes-instance of 3-partition, then $(T, k, m)$ is a yes-instance of PFET. Let $\{a_{10}, a_{11}, a_{12}\}, \cdots, \{a_{n0}, a_{n1}, a_{n2}\}$ be a partition of $A$ that satisfies $s(a_{i0}) + s(a_{i1}) + s(a_{i2}) = B$, $1 \leqq i \leqq n$, and consider the following $n + 1$ equal-sized, disjoint subgraphs of $T$. Start by putting $S_0$ into one subgraph and the triples $S_{i0}$, $S_{i1}$, $S_{i2}$ each into a separate subgraph; then take the one leaf that is adjacent to the root of $S_0$ from each of $S_{i0}$ and $S_{i1}$, $1 \leqq i \leqq n$, and move it to the subgraph containing $S_0$, in order to make all of the subgraphs the same size. These subgraphs can be filled out so that they are all isomorphic, and the edges between them can be filled out so that they present isomorphisms that are consistent. Thus, $T$ can be filled out to a product graph in which one factor has $n + 1$ nodes.

Conversely, suppose that $(T, k, m)$ is a yes-instance of PFET. Let $H = H_1 \square H_2$ be a factorable extension of $T$ with $|V(H_1)| = k = n + 1$, and fix a particular partition of $T$ into $n + 1$ copies of $H_2$. To each star $S_i$ in $T$, we associate a *home copy* of $H_2$—the copy that contains the root of $S_i$. In any product graph, a node in a particular copy of the right factor is adjacent to at most one node in each other copy. Thus, at most $n$ nodes of $S_i$ can lie outside of $S_i$'s home copy. This, together with the fact that each copy has $Bj_1 + 3j_2 + 3n + 1$ nodes, implies that no $S_i$, where $i > 0$, shares a home copy with $S_0$ and that each $S_i$, where $i > 0$, has at least $s(a_i)j_1 + j_2 + 1$ nodes in its home copy.

If four stars $S_{i0}$, $S_{i1}$, $S_{i2}$, and $S_{i3}$ have the same home copy, then we must have

$$\sum_{p=0}^{3} (s(a_{ip})j_1 + j_2 + 1) \leqq Bj_1 + 3j_2 + 3n + 1,$$

because there are $Bj_1 + 3j_2 + 3n + 1$ nodes in each copy. This cannot be, however, since $\sum_{0 \leqq p \leqq 3} s(a_{ip}) > B$ and $4j_2 + 4 > 3j_2 + 3n + 1$, because $j_2 = 3n$. Hence each copy of $H_2$ is associated with the distinguished star $S_0$ or with exactly three undistinguished stars. To finish the proof, we show that, if $S_{i0}$, $S_{i1}$, and $S_{i2}$ share a home copy, then $s(a_{i0}) + s(a_{i1}) + s(a_{i2}) = B$, which implies that there is a 3-partition that corresponds to this product graph.

It suffices to show that, for any three such stars, $s(a_{i0}) + s(a_{i1}) + s(a_{i2}) \leqq B$, because the sum over all $n$ triples has to be $nB$. Suppose that $s(a_{i0}) + s(a_{i1}) + s(a_{i2}) > B$, and that the associated stars share a home copy. Then the number of nodes in this copy is at least

$$\sum_{p=0}^{2} (s(a_{ip})j_1 + j_2 + 1) \geqq (B + 1)j_1 + 3j_2 + 3 = Bj_1 + 3j_2 + j_1 + 3.$$

We now have a contradiction: this number is greater than $Bj_1 + 3j_2 + 3n + 1$, the size of $H_2$, because $j_1 + 3 = 3n + 3 > 3n + 1$. □

Note that the PSFE problem and the problem of deciding whether a graph is a subgraph of the $k$-dimensional hypercube are unusual in that they are NP-complete graph problems that remain NP-complete when the input graph is constrained to be a tree [7].

For the rest of this section, we assume that our trees are arbitrarily rooted. $T$ will always denote a tree on $n$ nodes. $T$ can always be extended trivially to a product graph on $2n - 2$ nodes: divide $V(T)$ into two sets $V_1$ and $V_2$ by putting one leaf into $V_1$ and the rest of $V(T)$ into $V_2$, then fill out $V_1$ so that it is isomorphic to the subtree induced

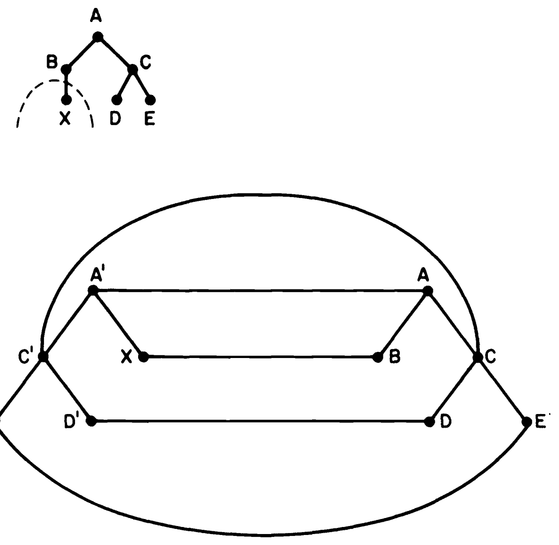


FIG. 11. *Extending a tree on n* = 6 *nodes to a product graph on* $2n - 2 = 10$ *nodes.*

by $V_2$ and draw in an isomorphism between the two copies (see Fig. 11). There are trees, e.g., the $n$-node stars, for which this is the smallest factorable extension. In what follows, we assume that all factorable extensions have at most $2n - 2$ nodes.

**5.1. Factorable extensions with $k = 2$.** Let $H \cong H_1 \square H_2$ be a factorable extension of $T$ with $|V(H_1)| = 2$. Our assumption that $|V(H)| \leqq 2n - 2$ implies immediately that $H_1 = K_2$. So we can view the process of extending $T$ to $H$ as follows. Divide $V(T)$ into two sets $V_1$ and $V_2$ such that no node in $V_1$ is adjacent to more than one node in $V_2$ and vice versa. Then add isolated nodes to the smaller of $V_1$ and $V_2$ so that $|V_1| = |V_2| = |V(H)|/2$, add edges to both node sets so that they induce isomorphic graphs, and draw in the isomorphism edges that were not already in $E(T)$. This is a simplified version of the process described in § 3 for general graphs.

Let $V_1 \sqcup V_2$ be any partition of $V(T)$. We call the set of edges of $E(T)$ with one endpoint in each $V_i$ the *cutset*. The following remark summarizes the principle we use in our dynamic programming algorithm.

*Remark* 5.1. A tree $T$ has a factorable extension of the form $T \subset K_2 \square H_2$ with $|V(H_2)| = m$ if and only if the tree can be partitioned into two vertex sets $V_1$ and $V_2$ such that the cutset is a matching and $m \geqq \max(|V_1|, |V_2|)$.

Thus we can solve the PFET problem if we can determine all possible pairs of values $(|V_1|, |V_2|)$ achievable by partitions for which the cutset is a matching.

The algorithm proceeds bottom-up. We associate with the root of each subtree a set of labels of the form $(x, y)^z$, where $x$ and $y$ are integers and $z$ is one of the letters $c$ and $u$, for *constrained* and *unconstrained*, respectively. If node $v$ has the label $(x, y)^c$, then we can partition the subtree rooted at $v$ into sets $V_1$ and $V_2$ such that $v \in V_1$, $|V_1| = x$, $|V_2| = y$, and there is an edge incident to $v$ in the cutset. If $v$ has the label, $(x, y)^u$, then there is a partition that satisfies the same conditions except that there is no edge incident to $v$ in the cutset. In both cases, there may be more than one such partition. Our algorithm computes the set of labels for $v$ from the sets of labels of its children by combining the partitions of its children's subtrees in all possible ways (as explained below). The significance of the *constraint* is that if an edge from $v$ to one of its children is in the cutset, then the edge from $v$ to its parent cannot be in the cutset, because this would violate our requirement that the cutset be a matching.

We now show how to construct the labels for $v$ given the labels for its children. Each leaf is labeled $(1, 0)^u$. Let $L(v_i)$ be the set of labels for the subtree rooted at $v_i$. If $v_1, \cdots, v_d$ are the children of $v_0$, then there are two ways to form partitions for $v_0$ out of those for $v_i$. First, if $(x_1, y_1)^{z_1} \in L(v_1), \cdots, (x_d, y_d)^{z_d} \in L(v_d)$, $x = 1 + x_1 + \cdots + x_d$, and $y = y_1 + \cdots + y_d$, then $(x, y)^u \in L(v_0)$ (see Fig. 12). Second, if $(x_1, y_1)^{z_1} \in L(v_1), \cdots, (x_d, y_d)^{z_d} \in L(v_d)$, $x = 1 + x_1 + \cdots + x_d - x_s + y_s$, $y = y_1 + \cdots + y_d - y_s + x_s$, and $z_s = u$ for some $s$, then $(x, y)^c \in L(v_0)$ (see Fig. 13). No other labels are in $L(v_0)$.

Hence, it is clear that we can compute $L(v_r)$, where $v_r$ is the root of our tree, in a bottom-up fashion. It remains to show that the computation can be performed in polynomial time. Note that the number of labels in any set $L(v)$ is trivially bounded above by $2(n - 1)$; so we never have to store more than polynomially much data. We use the following notation to complete the proof:

$$C(v_i) = \{(x,y) \mid (x,y)^c \in L(v_i)\},$$

$$U(v_i) = \{(x,y) \mid (x,y)^u \in L(v_i)\},$$

$$A(v_i) = C(v_i) \cup U(v_i),$$

$$U^R(v_i) = \{(y,x) \mid (x,y)^u \in L(v_i)\}.$$

If $A$ and $B$ are sets of ordered pairs, then we denote by $A + B$ the sum of the sets, i.e.,

$$A + B = \{(x_A + x_B, y_A + y_B) \mid (x_A, y_A) \in A, (x_B, y_B) \in B\}.$$

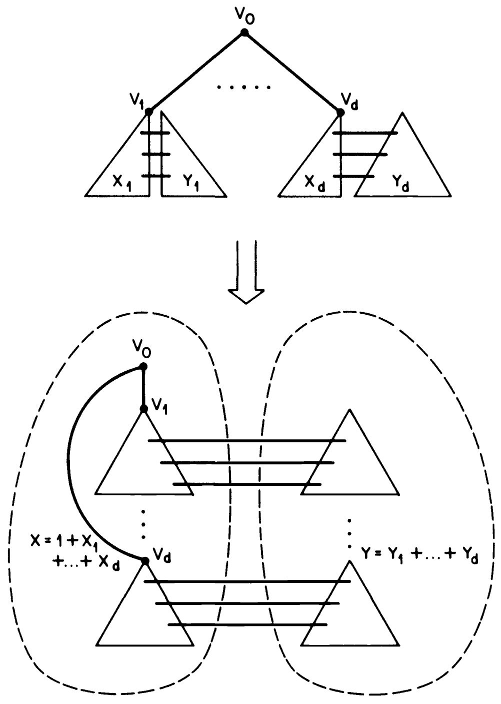
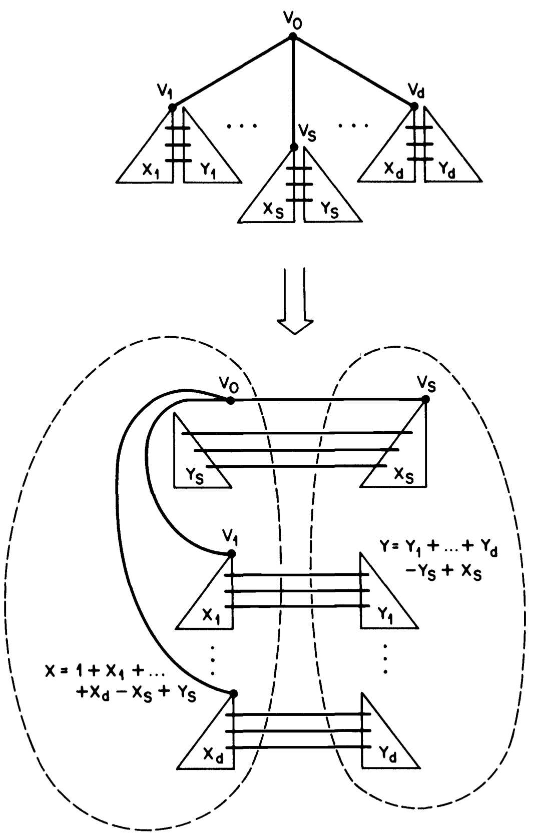


FIG. 12. *Combining the partitions corresponding to the children's labels* $(x_i, y_i)^{z_i}$ *into a partition corresponding to* $(1 + x_1 + \cdots + x_d, y_1 + \cdots + y_d)^u$. *The cutset is the union of the cutsets in the subtrees.*



$v_0$
$v_1$
$v_s$
$v_d$
$X_1$
$Y_1$
$X_s$
$Y_s$
$X_d$
$Y_d$
$Y = Y_1 + \dots + Y_d - Y_s + X_s$
$X = 1 + X_1 + \dots + X_d - X_s + Y_s$

FIG. 13. *Combining the partitions corresponding to* $(x_1, y_1)^{z_1}, \cdots, (x_s, y_s)^u, \cdots, (x_d, y_d)^{z_d}$ *into the partition* $(1 + x_1 + \cdots + x_d - x_s + y_s, y_1 + \cdots + y_d - y_s + x_s)^c$. *The cutset is the union of the cutsets in the subtrees plus the edge* $v_0 - v_s$.

If $v_1, \cdots, v_d$ are the children of $v_0$ then

$$U(v_0) = \{(1,0)\} + \sum_{i=1}^{d} A(v_i), \tag{6}$$

$$U^R(v_0) = \{(y,x) \mid (x,y) \in U(v_0)\}. \tag{7}$$

When we let

$$C_j(v_0) = \{(1,0)\} + U^R(v_j) + \sum_{i \neq j} A(v_i), \tag{8}$$

it follows that

$$C(v_0) = \bigcup_j C_j(v_0). \tag{9}$$

Because we can compute $A + B$ in time $O(|A| \cdot |B|)$, we can compute $C(v)$, $U(v)$, and $U^R(v)$ in time polynomial in the number of labels associated with $v$'s children. By our remark that there are only polynomially many labels in the whole tree, we know that we can find $A(v_r)$ in polynomial time. A tree $T$ with $2n$ nodes can be extended to a product graph $K_2 \,\Box\, G_2$ without adding any new nodes if and only if $(n, n) \in A(v_r)$. Thus we have the following theorem.

THEOREM 5.2. *The* PFET *problem can be solved in polynomial time if* $k = 2$.

**5.2. Binary trees.** The method of § 5.1 lets us decide for any tree $T$ on $2n$ nodes whether $T$ has a factorable extension of the form $T \subset H_1 \,\square\, H_2$ with $H_1 \cong K_2$ and $|V(H_2)| = n$. In this section, we show that if $T$ is a binary tree other than the four-node star $K_{1,3}$, then the answer to this question is *yes*. Our proof of this implies an $O(n)$ algorithm to find the right factor $\overline{H_2}$ of such an extension.

As noted above, an algorithmic statement of this problem is as follows: Given a binary tree with $2n$ nodes, partition its vertices into two sets $V_1$ and $V_2$ such that $|V_1| = |V_2| = n$ and the cutset is a matching.

For this proof, we use a modification of the labels used in § 5.1. Instead of $(x, y)^c$, we use the label $w^c$, where $w = x - y$; similarly we use the label $w^u$ instead of $(x, y)^u$.

As above, define the notation

$$C(v_i) = \{w \mid w^c \in L(v_i)\},$$

$$U(v_i) = \{w \mid w^u \in L(v_i)\},$$

$$A(v_i) = C(v_i) \cup U(v_i),$$

$$U^R(v_i) = \{-w \mid w^u \in L(v_i)\},$$

and the sum of two sets of numbers, $A$ and $B$

$$A + B = \{w_A + w_B \mid w_A \in A, w_B \in B\}.$$

We can rewrite equations (6)–(9) as follows (where $v_0$ is the root of a binary subtree and $v_l$ and $v_r$ are its left and right children):

$$U(v_0) = \{1\} + A(v_l) + A(v_r),$$

$$U^R(v_0) = \{-w \mid w \in U(v_0)\},$$

$$C(v_0) = (\{1\} + A(v_l) + U^R(v_r)) \cup (\{1\} + U^R(v_l) + A(v_r)).$$

Let $B(T_1, T_2)$ be the tree with a root whose left child is the rooted tree $T_1$ and whose right child is the rooted tree $T_2$. Then the set $S$ of rooted binary trees can be defined as follows. The empty tree (that is, the tree with no nodes and no edges) is in $S$; $B(T_1, T_2)$ is in $S$ for every $T_1$ and $T_2$ (not necessarily distinct) in $S$, and nothing else is in $S$. We wish to classify each $T$ in $S$ according to the set $L(\text{root}(T))$ of labels achievable by the root of $T$. We identify a class with a set of labels. Our goal is to place each tree with an even number of nodes (except $K_{1,3}$) in a class that has the label $0^c$.

The set of classes we define must be *complete*, by which we mean that each rooted binary tree must belong to a class, and it must be *closed*, by which we mean that if $T_1$ and $T_2$ both fall into one of our classes, then so does $B(T_1, T_2)$. The set of classes we use is $S_1 = \{0^c\}$, $S_2 = \{1^u\}$, $S_3 = \{0^c, 2^u\}$, $S_4 = \{0^c, 2^u, 4^u\}$, $S_5 = \{1^c, 3^u\}$, $S_6 = \{\pm 2^c, 4^u\}$, $S_7 = \{-1^u, \pm 3^c\}$, $S_8 = \{\pm 1^c, \pm 3^c\}$, $S_9 = \{0^c, \pm 2^c\}$, and $S_{10} = \{\pm 1^c, 3^u\}$. We limit the number of classes to ten by *dropping* and *downgrading* labels. Dropping $w$ from a class $S_i$ simply means that $w$ is achievable by $L(\text{root}(T))$ for some $T \in S_i$, but we choose to ignore it. Thus, we are making a weaker statement about the set of legal partitions of trees in $S_i$ than we might, but we have a set of classes that is easier to prove closed. Similarly, we downgrade a label $w^u$ by replacing it with $w^c$—if a subtree can achieve the label $w^u$, then we can certainly impose an unnecessary constraint on the edge from its root to the parent of its root and say that the subtree has achieved the label

$w^c$. A rooted binary tree $T$ is of type $i$ if $S_i$ is the highest-numbered class that can be obtained by computing $L(\text{root}\,(T))$ and dropping and downgrading labels.

We use the table Tab below to verify that our set of classes is both complete and closed. Interpret the table entries as follows: Tab $(r, c)$ is the table entry in the row labeled by class $r$ and the column labeled by class $c$; the table is symmetric, i.e., Tab $(r, c)$ = Tab$(c, r)$; if the children of the root of a tree $T$ are themselves roots of trees in classes $r$ and $c$, then $T$ is in class Tab $(r, c)$.

The proof that our set of classes is closed (that is, that Table 1 entries are correct) is straightforward, but tedious; we give two examples of the types of calculations that are necessary.

For $L(v_l) = \{0^c\}$, and $L(v_r) = \{0^c, 2^u\}$, we compute $L(v_0)$ as follows:

$$
\begin{aligned}
U(v_l) &= \{\},\\
C(v_l) &= \{0\},\\
U(v_r) &= \{2\},\\
C(v_r) &= \{0\},\\
U(v_0) &= \{1\}+\{0\}+\{0,2\}\\
&= \{1,3\},\\
C(v_0) &= (\{1\}+\{0\}+\{-2\}) \cup (\{1\}+\{\}+\{0,2\})\\
&= \{-1\}.
\end{aligned}
$$

This gives us $L(v_0) = \{-1^c, 1^u, 3^u\}$. After downgrading the $1^u$, we get $L(v_0) = \{\pm 1^c, 3^u\}$. Note that this proof of correctness of one specific table entry also yields a proof for all other table entries where $\{0^c\} \subseteq L(v_l)$ and $\{0^c, 2^u\} \subseteq L(v_r)$ or vice versa.

TABLE 1

| | $0^c$ | $1^u$ | $0^c, 2^u$ | $0^c, 2^u, 4^u$ | $1^c, 3^u$ |
|---|---|---|---|---|---|
| $0^c$ | $1^u$ | $0^c, 2^u$ | $\pm 1^c, 3^u$ | $\pm 1^c, 3^u$ | $\pm 2^c, 4^u$ |
| $1^u$ | | $1^c, 3^u$ | $0^c, 2^u, 4^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
| $0^c, 2^u$ | | | $\pm 1^c, 3^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $0^c, 2^u, 4^u$ | | | | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $1^c, 3^u$ | | | | | $\pm 1^c, 3^u$ |

| | $\pm 2^c, 4^u$ | $-1^u, \pm 3^c$ | $\pm 1^c, \pm 3^c$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
|---|---|---|---|---|---|
| $0^c$ | $-1^u, \pm 3^c$ | $0^c, \pm 2^c$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $1^u$ | $0^c, \pm 2^c$ | $\pm 1^c, \pm 3^c$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
| $0^c, 2^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $0^c, 2^u, 4^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $1^c, 3^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
| $\pm 2^c, 4^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $-1^u, \pm 3^c$ | | $\pm 1^c, 3^u$ | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
| $\pm 1^c, \pm 3^c$ | | | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ | $\pm 1^c, 3^u$ |
| $0^c, \pm 2^c$ | | | | $\pm 1^c, 3^u$ | $0^c, \pm 2^c$ |
| $\pm 1^c, 3^u$ | | | | | $\pm 1^c, 3^u$ |

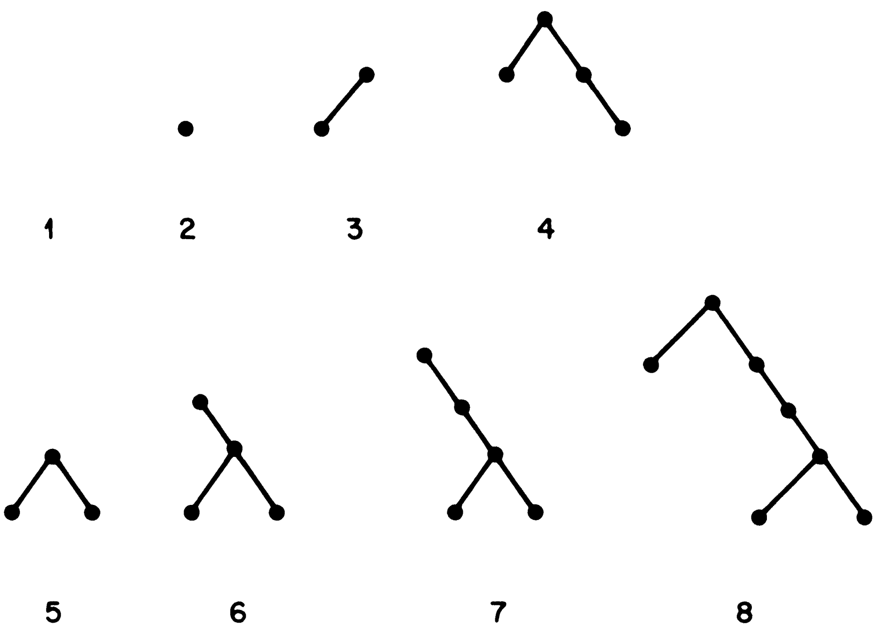


FIG. 14. *The eight binary trees that do not belong to $S_9$. Number* 1 *is the empty tree.*

For $L(v_l) = L(v_r) = \{1^c, 3^u\}$, the calculation goes as follows:

$$\begin{aligned} U(v_l) &= \{3\}, \\ C(v_r) &= \{1\}, \\ U(v_0) &= \{1\} + \{1,3\} + \{1,3\} \\ &= \{3,5,7\}, \\ C(v_0) &= \{1\} + \{1,3\} + \{-3\} \\ &= \{-1,1\}, \end{aligned}$$

yielding $L(v_0) = \{\pm 1^c, 3^u\}$ after dropping some labels.

Our proof of closure, along with the recursive definition of $S$ given above, also provides a proof of completeness: to classify a tree $T$, construct it recursively, looking up the appropriate table entry every time a new root is created (the empty tree is of type 1). A legal partition can also be created in linear time with this recursive algorithm.

Classes $S_1$–$S_8$ each contain only one tree (see Fig. 14). Intuitively, this is because all larger trees attain the additional flexibility of classes $S_9$ and $S_{10}$. Only one of the 10 classes has no label $0^c$ and also contains a tree with an even number of nodes—that is, class $S_6$ contains $K_{1,3}$. Aside from the trees in Fig. 14, all other trees with an even number of nodes belong to $S_9$, which has the label $0^c$. Therefore, we have the following theorem.

THEOREM 5.3. *Every binary tree $T \not\cong K_{1,3}$ with an even number of nodes has a factorable extension $H \cong K_2 \square H_2$, with $|V(H)| = |V(T)|$.*

Each binary tree with an odd number of nodes belongs to a class that contains one of the labels $\pm 1^c$, $\pm 1^u$. Thus we have an analogous result for this case.

THEOREM 5.4. *Every binary tree $T$ with an odd number of nodes has a factorable extension $H \cong K_2 \square H_2$, with $|V(H)| = |V(T)| + 1$.*

**5.3. Factorable extensions with $k = 3$.** The dynamic programming algorithm for PFET with $k = 3$ has the same structure as the one for $k = 2$; that is, we compute the set of triples of sizes $(x_1, x_2, x_3)$ achievable by legal partitions of $V(T)$ into three sets $(V_1, V_2, V_3)$. We do this bottom-up by combining, at each node, the legal partitions of the subtrees rooted at its children in all legal ways. The total amount of information stored at each node is polynomial and can be computed efficiently; thus the bottom-up traversal of the tree can be completed in polynomial time.

In the $k = 2$ case, the only requirement on the cutset was that it be a matching. Here we have more requirements, and the algorithm is accordingly more complicated. Let $V_1 \sqcup V_2 \sqcup V_3$ be a legal partition of $V(T)$ that corresponds to the node-numbering $\{(i, j)\}$, where $1 \leqq i \leqq 3$. (If this partition corresponds to a factorable extension $H$, where $|V(H)| = |V(T)|$, then, for each value of $i$, the second coordinate $j$ takes on the values one through $|V(T)|/3$. Otherwise, there will be a different number of $j$'s for each $i$.) The cutset is the set of edges $(i, j) — (i', j') \in E(T)$ in which $i \neq i'$. The first fact to observe is that the cutset cannot contain any paths of length three. Suppose it did contain such a path, say $(i_1, j_1) — (i_2, j_2) — (i_3, j_3) — (i_4, j_4)$. Because this is a legal numbering, we know that $j_1 = j_2 = j_3 = j_4$. At least two of $i_1$, $i_2$, $i_3$, and $i_4$ must be equal, thus creating a cycle, which cannot occur in a tree.

We partition the cutset $C$ into $P \sqcup P_{12} \sqcup P_{13} \sqcup P_{23}$. The set $P$ contains all paths of length two both of whose edges are in $C$. By definition of a cutset, each path in $P$ has one node from each of the vertex sets $V_i$. The set $P_{12}$ contains all edges $e = v_1 — v_2$ in $C$ such that $v_i \in V_i$ and $e$ is not half of a path in $P$. Similarly, $P_{13} = \{e = v_1 — v_3: v_i \in V_i\}$ and no path in $P$ contains $e$, and $P_{23}$ is defined analogously. Note that, by the definitions of $P_{ij}$ and the properties of a cutset, no node can lie on two paths in $C$; e.g., if $v — w$ is in $P_{12}$, then there is no $u$ for which $v — u$ is in $P_{13}$. Let $p = |P|$ and $p_{ij} = |P_{ij}|$. Then the counterpart of Remark 5.1 for the $k = 3$ case is the following.

*Remark* 5.2. A tree $T$ has a factorable extension of the form $T \subset K_3 \square G_2$, with $|V(G_2)| = m$, if and only if $V(T)$ can be partitioned into three sets $V_1 \sqcup V_2 \sqcup V_3$ such that the induced cutset $C = P \sqcup P_{12} \sqcup P_{13} \sqcup P_{23}$ has no paths of length three and $m \geqq p + p_{12} + p_{13} + p_{23}$.

As in the algorithm for the $k = 2$ case, we keep track of the legal partitions of the subtree rooted at a node $v$ with a set of labels $L(v)$, this time of the form $(x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23})^z$. If $L(v)$ contains this label, then there is a legal partition of the subtree rooted at $v$ with $|V_i| = x_i$ and cutset $C = P \sqcup P_{12} \sqcup P_{13} \sqcup P_{23}$, $|P| = p$, and $|P_{ij}| = p_{ij}$. If $z = u$, then there is no path in $C$ that contains $v$; if $z = o$, then there is a path in some $P_{ij}$ that contains $v$; if $z = t$, then $v$ is contained in a path in $P$. Without loss of generality, we assume that the root $v$ of the subtree is in $V_1$ and that if there is a node $w$ such that $v — w$ is in $P_{ij}$, then $w$ is in $V_2$.

Let $U(v) = \{(x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23}): (x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23})^u \in L(v)\}$. Define $O(v)$ and $T(v)$ analogously. If $A \subseteq U(v) \cup O(v) \cup T(v)$, then we use $S_{ij}(A)$ to denote the set obtained by taking each label in $A$, exchanging $x_i$ and $x_j$ and changing the $p_{kl}$'s appropriately; for example, $(x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23})$ in $A$ corresponds to $(x_2, x_1, x_3; p, p_{12}, p_{23}, p_{13})$ in $S_{12}(A)$. The sum $A + B$ of two sets of labels is $\{(x_1 + x'_1, x_2 + x'_2, x_3 + x'_3; p + p', p_{12} + p'_{12}, p_{13} + p'_{13}, p_{23} + p'_{23}): (x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23}) \in A$ and $(x'_1, x'_2, x'_3; p', p'_{12}, p'_{13}, p'_{23}) \in B\}$. Note that any subtree that can achieve a partition corresponding to $(x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23})^u$ can also achieve one corresponding to $(x_1, x_3, x_2; p, p_{13}, p_{12}, p_{23})^u$; hence $S_{23}(U(v_j)) = U(v_j)$.

It remains to give the recurrence equations corresponding to (6)–(9). As in the $k = 2$ case, the total number of labels that may be stored at each node and the time to perform the + and ∪ operations are polynomially bounded; hence the set $L(\text{root}(T))$ can be found in polynomial time. Let $v_1, \cdots, v_d$ be the children of $v_0$. Then

$$A(v_i) = (U(v_i) \cup O(v_i) \cup T(v_i)) \cup S_{23}(U(v_i) \cup O(v_i) \cup T(v_i)),$$

$$U(v_0) = \{(1,0,0;0,0,0,0)\} + \sum_{i=1}^{d} A(v_i),$$

$$O_j(v_0) = \{(1,0,0;0,1,0,0)\} + S_{12}(U(v_j)) + \sum_{i \neq j} A(v_i),$$

$$O(v_0) = \bigcup_j O_j(v_0),$$

$$T_j(v_0) = \{(1,0,0;1,0,0,-1)\} + S_{12}(S_{23}(O(v_j))) + \sum_{i \neq j} A(v_i),$$

$$T_{hj}(v_0) = \{(1,0,0;1,0,0,0)\} + S_{12}(U(v_h)) + S_{13}(U(v_j)) + \sum_{i \neq j, i \neq h} A(v_i),$$

$$T(v_0) = \left(\bigcup_j T_j(v_0)\right) \cup \left(\bigcup_{h \neq j} T_{hj}(v_0)\right).$$

The equation defining $T_j$ counts partitions represented by $(x_1, x_2, x_3; p, p_{12}, p_{13}, p_{23})^t$ that are formed by choosing a child $v_j$ of $v_0$ that can be the endpoint of a cut edge $v_j - w$, placing $w$ in $V_3$, placing $v_j$ in $V_2$, drawing in a cut edge from $v_0$ to $v_j$, and combining it with all the partitions achievable by the other children of $v_0$. Similar interpretations of the other equations are left as an exercise. Together with the previous discussion, they give this theorem.

THEOREM 5.5. *The* PFET *problem can be solved in polynomial time if* $k = 3$.

We conjecture that PFET can be solved in polynomial time using a similar algorithm for any fixed value of $k$. For successively larger values of $k$, the description of the allowable cutsets becomes much more complicated. This approach will not work for general graphs, because there is no simple way to combine partitions of subgraphs as there is to combine partitions of disjoint subtrees.

**6. Acknowledgment.** We are very grateful to our referee, who introduced us to the results of Wagner and suggested a way to simplify the proof of Theorem 3.1.

## REFERENCES

[1] S. BHATT, F. R. K. CHUNG, F. T. LEIGHTON, AND A. ROSENBERG, *Optimal simulation of tree machines*, in Proc. 27th Annual IEEE Symposium on Foundations of Computer Science, Toronto, Ontario, Canada, 1986.

[2] A. BRODER, D. DOLEV, M. FISCHER, AND B. SIMONS, *Efficient fault tolerant routings in networks*, Inform. Comput., 75 (1987), pp. 52–64.

[3] G. CYBENKO, D. W. KRUMME, AND K. N. VENKATARAMAN, *Hypercube embedding is* NP*-complete*, Technical Report 85-1, Tufts University, Medford, MA, 1985.

[4] J. FEIGENBAUM, *Product graphs: some algorithmic and combinatorial results*, Technical Report STAN-CS-86-1121, Ph.D. thesis, Stanford University, Stanford, CA, 1986.

[5] J. FEIGENBAUM, J. HERSHBERGER, AND A. A. SCHÄFFER, *A polynomial time algorithm for finding the prime factors of Cartesian product graphs*, Discrete Appl. Math., 12 (1985), pp. 123–138.

[6] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of* NP*-Completeness*, W. H. Freeman, San Francisco, 1979.

[7] D. S. JOHNSON, *The* NP*-completeness column: an ongoing guide*, J. Algorithms, 6 (1985), pp. 434–451.

[8] I. NIVEN AND H. S. ZUCKERMAN, *An Introduction to the Theory of Numbers*, 3rd ed., John Wiley, New York, 1972.

[9] G. SABIDUSSI, *Graph multiplication*, Math. Z., 72 (1960), pp. 446–457.

[10] A. S. WAGNER, *Embedding trees in the hypercube*, Technical Report 204/87, Ph.D. thesis, University of Toronto, Toronto, Ontario, Canada, 1987.

[11] P. WINKLER, *Factoring a graph in polynomial time*, European J. Combin., 8 (1987), pp. 209–212.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 219–230, May 1989

# RECURSIVE SOLUTIONS FOR THE GENERALIZED ERLANG QUEUEING SYSTEMS*

DALE R. FOX† AND S. KINGSLEY GNANENDRAN†

**Abstract.** An algorithm to compute the steady-state probabilities for the generalized Erlang queueing systems is presented. The algorithm is linear in the order of the number of states. The approach is based on applying the Matrix Tree Theorem to Markov Processes. The recursion is developed by considering the relationships between intrees to adjacent states. First, this is done for the generalized Erlang arrival system. The algorithm for the generalized Erlang service system follows easily by directional duality of the transition diagrams.

**Key words.** matrix tree theorem, intrees, semi-Markov processes, queueing

**AMS(MOS) subject classifications.** 60J27, 68R05, 90B22

## 1. Introduction.

Many queueing systems may be described with either deterministic or exponentially distributed interarrival and service times. These are the types of systems for which performance measures are most easily computed. Deterministic systems have input distributions that have a coefficient of variation of zero. The exponential distribution has a coefficient of variation that is equal to one. Many systems, particularly production systems, have input distributions that have a coefficient of variation between zero and one. The process times are not truly deterministic; yet, they are not as variable as the exponential. The Erlang distribution gives a method for modeling systems with inputs (interarrival or service times) that have coefficients of variation in this range.

Erlang distributions are also useful in modeling batch production, where a customer moving through a stage of an Erlang server corresponds to the service of a single item in a batch, or a customer moving through a stage of an Erlang arrival process corresponds to an arrival of an item in a batch that must wait for the whole batch to arrive before service starts. From these applications it is clear that it may often be convenient to allow the rate in each stage to depend on the stage and the number in the system. This will allow more flexibility in approximations, greater ability to explicitly model batch production where each item in a batch has a different processing time, and the ability to make the rates depend on the number of items already present. We consider such a generalization of the Erlang distribution. We also consider the case where the waiting space is limited. This is often more realistic than the assumption of unlimited waiting space.

This paper presents a recursive method for computing the steady-state probabilities for systems involving generalized Erlang distributions. The method has computational complexity that is linear in the order of the number of states of the system. The recursion is developed using the transition diagram of each system, the Matrix Tree Theorem for Directed Graphs, and certain kinds of subgraphs of the transition diagram called intrees. More will be said about this in the literature review. This recursive formulation may also be treated as a difference equation from which parametric solutions may be obtained. The effort involved will be related to the variability of the parameters and the size of the state space. The next section presents a formal statement of the model.

## 2. Problem statement.

This paper derives the steady-state distributions for two queueing systems. The first system is called the Generalized Erlang Arrival System, denoted

* Received by the editors May 4, 1988; accepted for publication (in revised form) September 7, 1988.

† Management Science Program, University of Tennessee, 617 Stokely Management Center, Knoxville, Tennessee 37996.

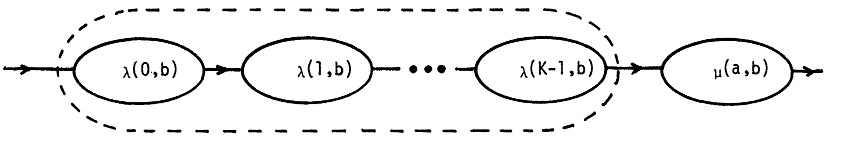


FIG. 1. *Network diagram for the Generalized Erlang Arrival System.*

$GE_K(a, b)/M(a, b)/./N + 1$, where the interarrival times are generalized Erlang, while the service times are exponential. A network diagram is given in Fig. 1. Note that the rates in the arrival process and the server depend on both the stage of the customer in the arrival process and the total number of customers in the system. In the arrival process this gives the potential for modeling systems with balking or blocking of arrival customers, systems requiring service to start only after a batch of customers arrive where the interarrival times for each item of the batch are independent exponential with rates that depend on which item in the batch it is and how many batches are already present, and greater flexibility in approximating systems whose interarrival times have coefficients of variation between zero and one. The service rate allows for the possibility of modeling multiple servers, and many other variations of state-dependent service rates where the rate depends on the state of the system, which is why the space for the number of servers has been filled with a period.

The second system considered is called the Generalized Erlang Service System, denoted $M(a, b)/GE_K(a, b)/./N + 1$. The interarrival times are exponential, while the service times are generalized Erlang. A network diagram of this system is contained in Fig. 2. Note the analogous state-dependence of the rates as in the generalized Erlang arrival system. The state-dependent arrival rate gives the potential for modeling systems with balking or blocking or many other features. The Generalized Erlang Service System may arise when processing a batch of customers with nonidentical service demands, where the rate also depends on the number of batches present. It is possible to model the presence of multiple servers by allowing the service rates to increase as the number in the system increases, but this will not be as exact as it was for the Generalized Erlang Arrival System. Both of these systems are finite-state, continuous-parameter Markov chains, and their steady-state distributions may be found by solving the steady-state equations.

The transition diagram of the Generalized Erlang Arrival System, $GE_K(a, b)/M(a, b)/./N + 1$, is given in Fig. 3. The states of the system are denoted by the ordered pair $(a, b)$ where $0 \leqq a \leqq K - 1$ and $0 \leqq b \leqq N$. $K$ denotes the number of stages, and $N$ denotes the system capacity. In the ordered pair $(a, b)$, the first entry stands for the number of stages of the arrival process that the arrival has completed, and the second entry represents the number in the system not in the arrival stages.

The Generalized Erlang Service System, $M(a, b)/GE_K(a, b)/./N + 1$, is represented by the transition diagram in Fig. 4. The states of the system are denoted by the ordered pair $(a, b)$ where $0 \leqq a \leqq K - 1$ and $0 \leqq b \leqq N$. The first entry represents the number

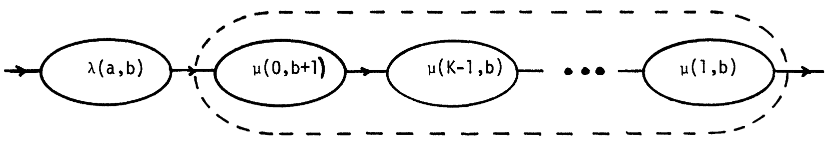


FIG. 2. *Network diagram for the Generalized Erlang Service System.*

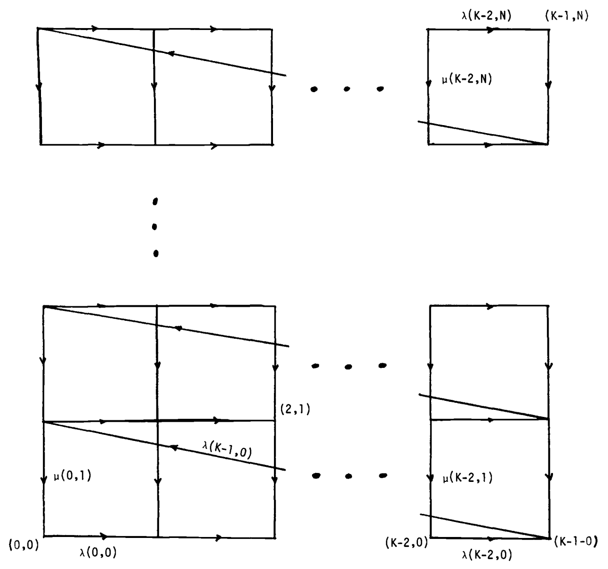


FIG. 3. *Transition diagram for the Generalized Erlang Arrival System.*

of service stages left to traverse for a customer in the service stages. The second entry denotes the number in the system that have not completed any service stages. The state $(0, b + 1)$ is the same as $(K, b)$. As in the previous system, the rates of the next event depend on both the number in the system and the stage of the system.

An interesting correspondence appears between the two transition diagrams, which will be called *directional duality*. By reversing the directions on the arcs of one of the transition diagrams and interchanging the weights λ and μ for each other, the transition diagram of the other system is obtained. Due to this directional duality, once the recursive solution is developed for one of these systems, the recursion for the other can be easily derived by a reverse substitution of the parameters.

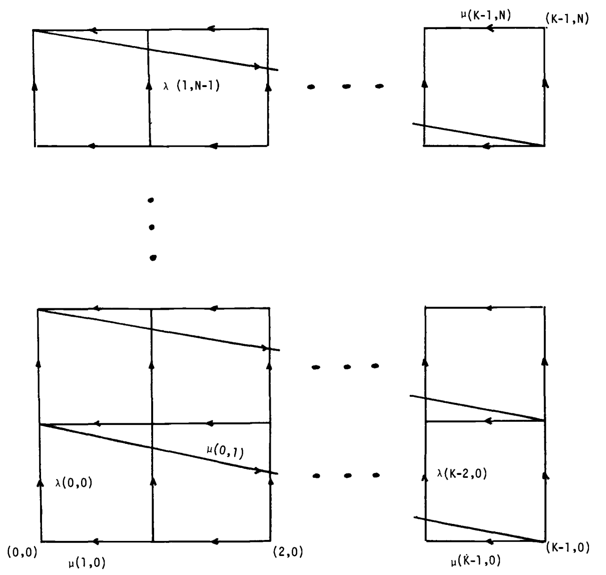


FIG. 4. *Transition diagram for the Generalized Erlang Service System.*

**3. Literature review.** Kleinrock [11] contains a discussion of the basic Erlang queueing systems. It contains a treatment that relies on transform analysis, but it is not usually possible to re-invert these transforms to obtain steady-state probabilities. Chaudhry and Templeton [3] also contains transform analysis of some generalizations of these systems as they relate to bulk queues.

A matrix method that works with the Erlang systems where the rates depend only on the stage and not on the number in the system is contained in Fox [6]. Fox [6] is a generalization of Disney [5] and makes use of eigenvalue decomposition of the matrices to get closed form solutions. In general, eigenvalue decomposition is less amenable to numerical computation than matrix inversion that has been used in more recent papers.

A fairly general method that treats the usual Erlang systems as a special case of the phase distributions is given in Neuts [12], [13]. This matrix-geometric approach will work on the usual generalization of the Erlang systems where the rates depend on the stage, but it will not allow the rate to depend on both the stage and the number in the system as in this paper. This eliminates some of the modeling capability from the problem. The approach exploits the block tridiagonal structure of the infinitesimal generator, and it is often more useful than Gaussian elimination because of its applicability when dealing with infinite queue lengths. As we have described the problem there are $K(N+1)$ states. The technique requires the construction and inversion of a $K \times K$ matrix for the $M/E_K/1/N+1$ case, and the inversion of $c$ different $K \times K$ matrices for the $E_K/M/c/N+c$. These inversions are also required for the systems with unbounded buffers. Matrix inversions have a number of multiplications that are on the order of the cube of the size of the matrix. The first, second, $\cdots$ $N$th powers of a $K \times K$ inverse for the $M/E_K/1/N+1$ case is required, which is on the order of $K^3$ at each step. The sequence of inverses obtained in the $E_K/M/c/N+c$ case must also be multiplied sequentially. This is most of what must be done to find an unnormalized solution to the steady-state equations. The method presented in Neuts [13] for normalization would require taking the sum of these sequential multiplications and multiplying them with some other vectors to obtain an initial probability vector, which must then be multiplied by the sequence of multiplication of matrices performed earlier to obtain the normalized probabilities. It is difficult to give a complete assessment of the order of magnitude of the computation because there are many different steps that must be combined; neither is one given in Neuts [13] where these results are presented.

The results in this paper build on the work contained in Solberg [14]. Solberg [14] gives a way to compute steady-state probabilities for a Markov process using the transition diagram of the process. This enhances the possibility of exploiting the structure of the Markov process directly without trying to mimic the structure in matrices, which may be impossible. This approach is an application of what is known as the Matrix Tree Theorem for Directed Graphs. A good reference containing results about this theorem is Harary and Palmer [10]. The papers by Chaiken and Kleitman [1] and Chaiken [2] contain alternative statements of the theorem and graph theoretic and combinatorial proofs. Because the theorem is usually used in studying Markov processes (for example, in Solberg [14] or Fox [8]), it requires the enumeration and sum of the weights of all the intrees to each state. Fortunately, we are able to avoid directly enumerating the intrees to any state. We develop a recursion that relates the sum of the weights of the intrees to one state to those surrounding it to get a more efficient solution. Using this approach we are also able to avoid the computational complexity and potential roundoff errors associated with matrix inversion and multiplication, and obtain maximum generality in the rate specifications.

**4. A recursive solution.** This section will start with a presentation of relevant theory and proofs. A series of results will first be developed that lead to the main result of the computation of the steady-state probabilities. An *intree to root i*, denoted $T_i$, of a strongly connected digraph $D$ is a subgraph of $D$ for which every vertex but $i$ has exactly one arc emanating from it, $i$ has none, and the underlying nondirected graph is circuit-free. Strong connectivity of a transition diagram is equivalent to irreducibility of the corresponding Markov process. The queueing systems studied here are irreducible and aperiodic, and since they are finite, we know the limiting distribution exists. One important consequence of this definition is that an intree rooted at $i$ must include a path from any vertex to $i$.

INTREE PATH LEMMA. *An intree to a root i of a strongly connected digraph D contains a unique directed path from any point to i.*

This lemma is a direct result of a standard equivalent definition for an intree; thus, the proof will be omitted. Harary [9] and Fox [7] contain other consequences of the definition and equivalent definitions for intrees. These references also contain definitions of the more basic concepts used above to define intrees. The *weight* $w(T_i)$ of an intree $T_i$ rooted at $i$ is defined as the product of the weights of its arcs. Let

$$\mathcal{T}_i = \{ T_i \mid T_i \text{ is an intree to } i \}$$

be the collection of all intrees to state $i$.

MATRIX TREE THEOREM FOR MARKOV PROCESSES. *An unnormalized solution for state i to the steady-state equations of a finite, irreducible, continuous-parameter Markov process is given by*

$$u_i = \sum_{T_i \in \mathcal{T}_i} w(T_i)$$

*where the sum is over all intrees to the vertex i in the transition diagram of the Markov process.*

This theorem is proved in Solberg [14] with a different but equivalent definition for intrees. This theorem will be the basis for the approach taken in this paper. Also remember that, since we are dealing with finite-state Markov processes, the steady-state probabilities may be easily obtained by normalizing the solution obtained from the theorem.

A subgraph of a transition diagram will be called a *function* if the subgraph is such that every state of the subgraph has exactly one emanating arc. A subgraph of a transition diagram will be called an $i$-function if the subgraph is such that every state other than $i$ in the transition diagram has exactly one emanating arc and $i$ has none. Note that a function may be defined over part of the transition diagram, while the $i$-function is defined over the whole transition diagram. From the definition of an intree stated earlier, it is apparent that an intree rooted at $i$ is the same as a circuit-free $i$-function. An equivalent statement of the Matrix Tree Theorem for Markov Processes is given in the following theorem, but first we define some notation.

$w_{ij}$ is the weight on arc $(i, j)$ in the transition diagram.

$\mathcal{C}_i = \{ C_i \mid C_i \text{ is an } i\text{-function that contains a directed circuit} \}$ is the collection of all $i$-functions containing directed circuits.

$w_{jl_j}$ is the weight on arc $(j, l_j)$ in some $i$-function containing a directed circuit.

CIRCUIT-FREE *i*-FUNCTION THEOREM.

$$u_i = \prod_{j \neq i} \left( \sum_k w_{jk} \right) - \sum_{C_i \in \mathscr{C}_i} \left( \prod_{j \neq i} w_{jl_j} \right)$$

*where $u_i$ is the unnormalized solution for the steady-state equations for state $i$. In the first term, the product term is over all states other than i, and the summation is over all states. In the second term, the summation is over all i-functions that contain a directed circuit, and the product is over all states other than $i$.*

*Proof.* This theorem takes the sum of the weights of all *i*-functions of the transition diagram and cancels out those containing a circuit to get the unnormalized solution, or the sum of the weight of the intrees to state $i$. The first term in the proposed solution is the sum of the weights of all *i*-functions. This term is a generating function where each term in the product is the sum of weights of the arcs emanating from a vertex other than the root $i$. The overall product then gives the sum of the weights of all the *i*-functions. However, *i*-functions are intrees rooted at *i* only when they do not contain a directed circuit. By subtracting from this quantity all the terms that contain directed circuits, we are left with the sum of the weights of the intrees. The second term is the one required to negate the presence of the *i*-functions that contain circuits. □

When stating the definitions and theorems previous to this point we have assumed that a state could be represented by a single variable $i$. Because the Generalized Erlang Systems require an ordered pair $(i, j)$ as the state to model the system as a Markov process, we will use $(i, j)$ as the state specification from now on. Due to the directional duality of the two Generalized Erlang Systems, all of the results that follow can be easily applied to either system. We will develop the results for the Generalized Erlang Arrival System and then use directional duality to get solutions for the Generalized Erlang Service System.

Given a state $(i, j)$ of the Generalized Erlang Arrival System, the *level* of the state is $j$. The state $(i, j)$ is *below* state $(h, k)$ if $j < k$, or if $j = k$ and $i < h$. This amounts to a reverse lexicographic ordering of the states, so that the states of the process are totally ordered. If the state $(i, j)$ is below $(h, k)$, then there is a unique path from $(i, j)$ to $(h, k)$, and the path contains all the states between them. The state $(h, k)$ is *above* $(i, j)$ if $(i, j)$ is below $(h, k)$. Any state that is above $(i, j)$ is separated from the states below $(i, j)$ by $(i, j)$, in the sense that if $(i, j)$ is removed there is no path from states below $(i, j)$ to states above $(i, j)$. This is not necessarily true for separating states below $(i, j)$ from states that are above. The unique path that connects the states below $(i, j)$ to the state $(i, j)$ is called the *arrival path*.

GENERALIZED ERLANG ARRIVAL SYSTEMS CIRCUIT LEMMA. *For the transition diagram of the Generalized Erlang Arrival System, all of the circuits are of length $K + 1$, there are KN distinct circuits, and for each state $(i, j)$, $(i, j)$ is lowest on no circuit if it is on the top row of the transition diagram.*

Fox [7] contains a formal proof, but the result is fairly obvious from looking at the transition diagram of the system.

GENERALIZED ERLANG ARRIVAL SYSTEM INTREE COMPOSITION LEMMA. *In the transition diagram of the Generalized Erlang Arrival System, a subgraph H is an intree rooted at state $(i, j)$ if and only if H can be formed from the superposition of the arrival path below $(i, j)$ and a circuit-free function on the states above $(i, j)$.*

*Proof.* $\Leftarrow$. The superposition of the arrival path to $(i, j)$ and a circuit-free function on the states above $(i, j)$ is a circuit-free $(i, j)$-function and, therefore, an intree to $(i, j)$.

$\Rightarrow$. Assume there is an intree rooted at $(i, j)$ that is not of this form. This implies there is a circuit-free $(i, j)$-function that is not of this form. Any circuit-free $(i, j)$-function may be looked at as the circuit-free superposition of a function on the states below $(i, j)$ and a function on the states above $(i, j)$. Since the superposition of these functions is circuit-free, each of the components must be circuit-free. The construction process allows for any circuit-free function on the states above $(i, j)$; thus, the intree that does not fit the desired form must contain a circuit-free function on the states below $(i, j)$ that is not the arrival path. Since state $(i, j)$ separates the states above $(i, j)$ from the states below $(i, j)$, none of the emanating arcs from states below $(i, j)$ can enter a state above $(i, j)$. But from the Intree Path Lemma, there must be a path from each state below $(i, j)$ to $(i, j)$, so that this circuit-free function on the states below $(i, j)$ must have such a path for each state. There is, however, only one such path, the arrival path; therefore, this circuit-free function on the states below $(i, j)$ cannot exist. □

Given state $(i, j)$ in the transition diagram of the Generalized Erlang Arrival System, the triple $(i, j, k)$, $k \geqq 0$, denotes the state that is $k$ states above $(i, j)$ on the arrival path. Thus, $(i, j)$ and $(i, j, 0)$ are the same state. $T(i, j, k)$ is the sum of the weighted intrees to the state that is $k$ states above $(i, j)$ on the arrival path. $T(i, j) = T(i, j, 0)$ is an unnormalized solution to the steady-state equations of the queueing system. $SW(i, j, k)$ denotes the sum of the weights of functions on states above $(i, j, k)$. $SWCF(i, j, k)$ denotes the sum of the weights of the circuit-free $(i, j, k)$ functions. $SWCFA(i, j, k)$ denotes the sum of the weights of the circuit-free functions on states above $(i, j, k)$. $WA(i, j, k)$ denotes the weight of the arrival path to state $(i, j, k)$. States $(i, N)$ where $i = 0, 1, \cdots, K-1$ and the state $(K-1, N-1)$ are called *upper boundary states* in the transition diagram. The states $(i, 0)$ where $i = 0, 1, \cdots, K-2$ are called the *lower boundary states*. All of the other states are called *interior states*.

RECURSION FOR THE STEADY-STATE PROBABILITIES OF THE GENERALIZED ERLANG ARRIVAL SYSTEM. *A set of unnormalized solutions to the steady-state equations of the Generalized Erlang Arrival System is*:

*Recursion for the upper boundary states*:

(I) $T(K-1, N) = 1$;

(II) $T(K-2, N) = \mu(K-1, N)/\lambda(K-2, N)$

(III) $T(i, j) = ((\lambda(i, j, 1) + \mu(i, j, 1))/\lambda(i, j))T(i, j, 1)$.

*Recursion for the interior states*:

$$\text{(IV)} \quad T(i,j) = \frac{\lambda(i,j,1)+\mu(i,j,1)}{\lambda(i,j)}\, T(i,j,1) - \frac{\mu(i,j,K+1)}{\lambda(i,j)}\, T(i,j,K+1).$$

*Recursion for the lower boundary states*:

$$\text{(V)} \quad T(i,j) = \frac{\lambda(i,j,1)}{\lambda(i,j)}\, T(i,j,1) - \frac{\mu(i,j,K+1)}{\lambda(i,j)}\, T(i,j,K+1).$$

*Proof.* In general, for all of the equations given above, we know from the Circuit-Free $i$-Function Theorem that an unnormalized solution to the steady-state equations for state $(i, j)$ can be obtained by taking the sum of the weights of all the $(i, j)$-functions and subtracting the weights of the $(i, j)$-functions containing circuits. Alternatively, we can find these unnormalized solutions by taking the sum of the weights of the intrees to $(i, j)$, from the Matrix Tree Theorem for Markov Processes. From the Generalized Erlang Arrival System Intree Composition Lemma, we also know that any of the intrees to a state $(i, j)$ can be formed from the superposition of the arrival path to $(i, j)$ and a circuit-free function on the states above $(i, j)$.

*Equation* (I). Since we are dealing with an unnormalized solution we choose to set the unnormalized value for the highest state equal to one.

*Equation* (II). For the state $(K-2, N)$ there is only one intree to the state since there is only one $(i, j)$-function on the state above it. The weight of this intree differs from the weight of the intree to the state above it by replacing $\mu(K-1, N)$ for $\lambda(K-2, N)$. Thus we get

$$T(K-2,N)=\frac{\mu(K-1,N)}{\lambda(K-2,N)}T(K-2,N,1).$$

However, $T(K-2, N, 1) = T(K-1, N) = 1$ and (II) is proved.

*Equation* (III). This recursion is for the remaining upper boundary states. From Fig. 5 and the Generalized Erlang Arrival Systems Circuits Lemma it is apparent that all of the functions on states above an upper boundary state are circuit-free. Because of the Generalized Erlang Arrival System Intree Composition Lemma and the preceding observation

(III.1) $$SWCF(i,j)=WA(i,j)SW(i,j).$$

The weight of the arrival path to $(i, j)$ is

(III.2) $$WA(i,j)=\frac{WA(i,j,1)}{\lambda(i,j)},$$

since it has one less arc than the arrival path to $(i, j, 1)$. The sum of the weights of the functions on the states above $(i, j)$ is

(III.3) $$SW(i,j)=[\lambda(i,j,1)+\mu(i,j,1)]SW(i,j,1),$$

since one of the arcs with weight $\mu(i, j, 1)$ or $\lambda(i, j, 1)$ must be included with each function on the states above $(i, j, 1)$. Substituting (III.2) and (III.3) into (III.1), we obtain

(III.4) $$SWCF(i,j)=\frac{\lambda(i,j,1)+\mu(i,j,1)}{\lambda(i,j)}WA(i,j,1)SW(i,j,1).$$

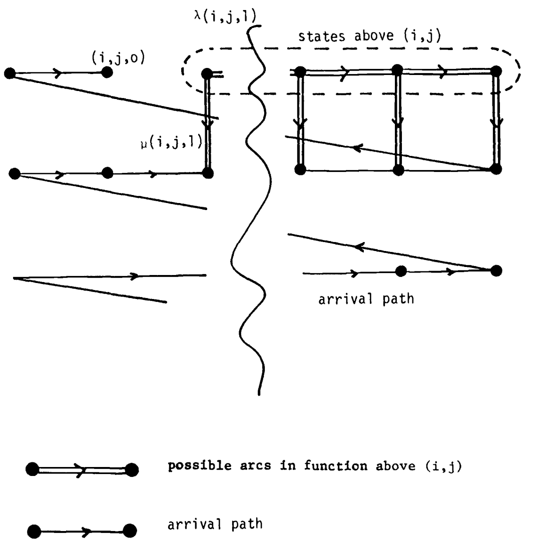


FIG. 5. *Diagram illustrating the decomposition for the upper boundary.* ●⟹● *possible arcs in function above* $(i, j)$; ●→● *arrival path.*

However, the left-hand side of the equality is $T(i, j)$, and the product of the last two terms on the right-hand side is $T(i, j, 1)$; therefore,

$$T(i,j)=\frac{\lambda(i,j,1)+\mu(i,j,1)}{\lambda(i,j)}T(i,j,1).$$

Thus, starting with the topmost state, we are now able to compute the unnormalized solution for each of the upper boundary states by working down from the top recursively.

*Equation* (IV). For the interior states we must account for the presence of circuits in the construction process. We continue with the approach of working recursively down the arrival path. Assume we have $T(i, j, 1)$ and all of the other unnormalized solutions for states above $(i, j)$. Remember that

(IV.1) $$T(i,j)= WA(i,j)SWCFA(i,j).$$

The second term in the product in (IV.1) is

(IV.2) $$\begin{aligned}&[\lambda(i,j,1)+\mu(i,j,1)]SWCFA(i,j,1)\\&-\lambda(i,j,1)(\text{Sum of weights of circuit-free functions above } (i,j,1)\\&\quad\text{for which including arc with } \lambda(i,j,1) \text{ forms a circuit})\\&-\mu(i,j,1)(\text{Sum of weights of circuit-free functions above } (i,j,1)\\&\quad\text{for which including arc with } \mu(i,j,1) \text{ forms a circuit}).\end{aligned}$$

However, including the arc with weight $\mu(i, j, 1)$ will not form any circuits with a circuit-free function on the states above $(i, j, 1)$. Thus the third term in (IV.2) is zero. Including the arc with weight $\lambda(i, j, 1)$ will cause exactly one circuit to be formed when it is included with many of the circuit-free functions on the states above $(i, j, 1)$. This circuit is formed whenever a circuit-free function on the states above $(i, j, 1)$ contains the arcs with weights $\lambda(i, j, 2), \cdots, \lambda(i, j, K)$ and $\mu(i, j, K + 1)$, regardless of the other arcs included in the function. This is illustrated in Fig. 6. Thus the second term in (IV.2) is the same as

$$\lambda(i,j,1)\cdots\lambda(i,j,K)\mu(i,j,K+1)SWCFA(i,j,K+1).$$

Putting this back into (IV.2) and then substituting (IV.2) into (IV.1), we obtain

(IV.3) $$\begin{aligned}T(i,j)&= WA(i,j)[\lambda(i,j,1)+\mu(i,j,1)]SWCFA(i,j,1)\\&\quad- WA(i,j)(\text{Weight of arrival path from } (i,j,1) \text{ to } (i,j,K+1))\\&\quad\cdot SWCFA(i,j,K+1)\mu(i,j,K+1).\end{aligned}$$

For the first term in the first product in (IV.3) we have

(IV.4) $$WA(i,j)=\frac{WA(i,j,1)}{\lambda(i,j)}.$$

Putting this back into the first product in (IV.3), we obtain

(IV.5) $$\frac{\lambda(i,j,1)+\mu(i,j,1)}{\lambda(i,j)}T(i,j,1)$$

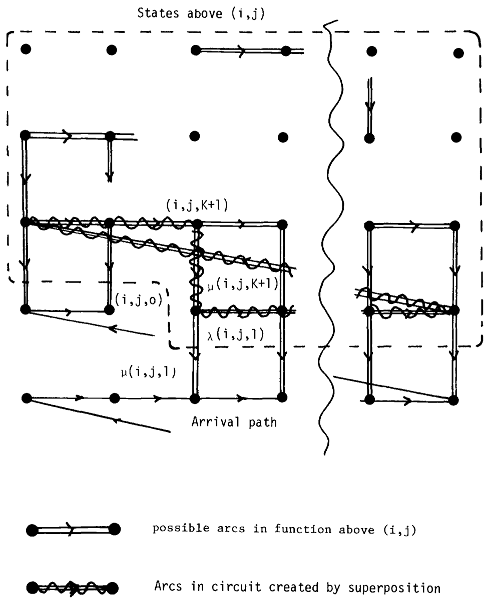

States above (i,j)
(i,j,K+1)
μ(i,j,K+1)
(i,j,o)
λ(i,j,1)
μ(i,j,1)
Arrival path
possible arcs in function above (i,j)
Arcs in circuit created by superposition

FIG. 6. *Diagram illustrating the decomposition for the interior states.* ●⟹● *possible arcs in function above* $(i, j)$; ●∿● *arcs in circuit created by superposition.*

as the first term in (IV.3) by the Generalized Erlang Arrival System Intree Composition Lemma. The second product in (IV.3) is

$$\text{(IV.6)} \qquad \frac{WA(i,j,K+1)}{\lambda(i,j)} SWCFA(i,j,K+1)\mu(i,j,K+1).$$

This is true, since the first two terms in the product would compose the arrival path to $(i, j, K+1)$, except for the arc with weight $\lambda(i, j)$. However, this is the same as

$$\text{(IV.7)} \qquad \frac{\mu(i,j,K+1)}{\lambda(i,j)} T(i,j,K+1)$$

due to the Generalized Erlang Arrival System Intree Composition Lemma. Putting (IV.5) and (IV.7) into (IV.3), we obtain (IV).

*Equation* (V). The lower boundary states are the same as the interior states except that the sum of the weights of the functions on the states above $(i, j)$ are not going to have any arcs with weight $\mu(i, j, 1)$. Thus the recursion and the proof is the same as for the interior states, but $\mu(i, j, 1) = 0$. □

Note that the most complicated recursion is for the interior states and that the recursions for the upper and lower boundaries are simplifications of this. These unnormalized solutions can readily be normalized by accumulating the values at each step of the recursion and then using this finite sum as the normalizing constant.

To obtain results for the Generalized Erlang Service System we need only exploit the directional duality between the two systems. The states of the Service System are ordered as they are for the Arrival System. The idea of separation is used differently in that now any state below $(i, j)$ is separated from a state above it by the removal of $(i, j)$. The unique path connecting any state above $(i, j)$ to those below $(i, j)$ is called the *departure path*. The triple $(i, j, k')$ denotes the state that is $k$ states below $(i, j)$ on the departure path. The lower boundary states of the Service System are the states $(i, 0)$ for $i = 0, 1, \cdots, K-1$ and $(0, 1)$. The upper boundary states are $(i, N)$ for $i = 1, \cdots, K-1$. The rest are interior states.

RECURSION FOR THE STEADY-STATE PROBABILITIES OF THE GENERALIZED ERLANG SERVICE SYSTEM. *A set of unnormalized solutions to the steady-state equations of the Generalized Erlang Service System is*:

*Recursion for the lower boundary states*:

(I) $T(0,0) = 1$;

(II) $T(1,0) = \lambda(0,0)/\mu(1,0)$;

(III) $T(i,j) = ((\mu(i,j,1') + \lambda(i,j,1'))/\mu(i,j))T(i,j,1')$.

*Recursion for the interior states*:

$$\text{(IV)}\quad T(i,j) = \frac{\mu(i,j,1') + \lambda(i,j,1')}{\mu(i,j)} T(i,j,1') - \frac{\lambda(i,j,(K+1)')}{\mu(i,j)} T(i,j,(K+1)').$$

*Recursion for the upper boundary states*:

$$\text{(V)}\quad T(i,j) = \frac{\mu(i,j,1')}{\mu(i,j)} T(i,j,1') - \frac{\lambda(i,j,(K+1)')}{\mu(i,j)} T(i,j,(K+1)').$$

*Note that the recursion works from the bottom to the top, the reverse of the recursion for the Arrival System.*

*Proof.* Take the transition diagram of the Service System, and rotate it $\pi$ radians in the plane. Substitute the weights on the arcs of the Service System for weights that correspond to the arcs in the Arrival System. The rotation implies that $(i, j, k')$ is used instead of $(i, j, k)$. $\square$

**5. Conclusions.** Many useful special cases are immediate consequences of the recursions for the Arrival and Service Systems. To obtain results for the Erlang queueing systems studied in elementary queueing texts let $\lambda(i,j) = K\lambda$ and $\mu(i,j) = \mu$ for the Arrival System, and let $\lambda(i,j) = \lambda$ and $\mu(i,j) = K\mu$ for the Service System. By letting $\lambda(i,j) = K\lambda$ and $\mu(i,j) = j\mu$ for $j \leqq m$, the steady-state probabilities for the multiple server case of the Arrival System described above may be obtained. One obvious advantage of the recursions given for the Generalized Erlang Queueing Systems is the generality available in the rate specifications.

The recursions may be altered to allow for enumerating the intrees to each state. They may also be used to get closed-form solutions in cases where the rates have some regularity. Essentially the same proof technique could be used with even more general distributions. For example, if one of the input distributions is Coxian and the other is exponential, then the same style of recursion and proof will work, though it will require more bookkeeping. Although this particular proof will be restricted to transition diagrams with special connectivity structure, the limits of this graph theoretic method have yet to be explored. Finally, although the results have been developed for the case of finite waiting spaces, it is apparent that, at least in the case of the Generalized Erlang Service System, closed-form solutions could be developed for systems with infinite waiting space where the transition rates are regular.

## REFERENCES

[1] S. CHAIKEN AND D. KLEITMAN, *Matrix tree theorems*, J. Combin. Theory Ser. A, 24 (1978), pp. 377–381.

[2] S. CHAIKEN, *A combinatorial proof of the all minors matrix tree theorem*, SIAM J. Algebraic Discrete Meth., 3 (1982), pp. 319–329.

[3] M. Chaudhry and J. Templeton, *A First Course in Bulk Queues*, Wiley-Interscience, New York, 1983.

[4] E. Cinlar, *Introduction to Stochastic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[5] R. Disney, *Matrix solution for the two-server queue with overflow*, Management Sci., 19 (1972), pp. 254–265.

[6] D. Fox, *The geometry of position in queueing networks*, M.S. thesis, Department of Industrial Engineering and Operations Research, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1979.

[7] ———, *Parametric and algorithmic solutions for the steady-state analysis of Markovian queueing systems using graphical enumeration*, Ph.D. dissertation, Department of Industrial Engineering, Purdue University, W. Lafayette, IN, 1984.

[8] ———, *Block cutpoint decomposition for Markovian queueing systems*, Appl. Stochastic Models and Data Anal., 4 (1988), pp. 101–114.

[9] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

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

[11] L. Kleinrock, *Queueing Systems*, John Wiley, New York, 1975.

[12] M. Neuts, *Matrix-Geometric Solutions in Stochastic Models*, The Johns Hopkins University Press, Baltimore, MD, 1981.

[13] ———, *Explicit steady-state solutions to some elementary queueing models*, Oper. Res., 30 (1982), pp. 480–489.

[14] J. Solberg, *A graph theoretic formula for the steady-state distribution of finite Markov processes*, Management Sci., 21 (1975), pp. 1040–1048.

[15] ———, *Some graph theoretic formulas for Markov processes*, Technical Report, School of Industrial Engineering, Purdue University, W. Lafayette, IN; Oper. Res., 23 (1975), pp. 594–602.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 231-239, May 1989

# DETERMINING RESISTANCES FROM BOUNDARY MEASUREMENTS IN FINITE NETWORKS*

GREGORY F. LAWLER†‡ AND JOHN SYLVESTER†§

**Abstract.** Small perturbations from a fixed resistance $\gamma$ in a finite electrical network can be determined from boundary measurements if every function on edges in the network is a linear combination of products of gradients of $\gamma$ harmonic functions. This condition holds for any finite subgraph of the cubic lattice $\mathbb{Z}^d$ and $\gamma = 1$.

**Key words.** electrical networks, inverse problems, discrete harmonic functions

**AMS(MOS) subject classifications.** 94C15, 60J27

**1. Introduction.** The current flow through a resistor and the voltage drop across the resistor are related by Ohm's Law:

$$V_l - V_r = IR. \tag{1.1}$$

$V_l$ and $V_r$ are the voltage potentials at the left and right leads, respectively; $I$ is the current flow from left to right; and $R$ is the resistance. If we measure both the voltage drop and the current, we may use (1.1) to determine $R$.

Now suppose we are given two resistors, wired in series and encased in a black box with only one lead from each resistor accessible (see Fig. 1). It is easy to check that we cannot determine the individual resistances $R_1$ and $R_2$ experimentally (i.e., from voltage and current measurements at the accessible leads). Indeed, only the sum, $R_1 + R_2$, can be computed.

However, for the two-dimensional analogue of this circuit (see Fig. 2), it is possible to find all the resistances experimentally. It requires two experiments. If, for example, we set the voltages $V_1^1 = 1$, $V_2^1 = V_3^1 = V_4^1 = 0$ in the first experiment and set $V_2^2 = 1$, $V_1^2 = V_3^2 = V_4^2 = 0$ in the second, labeling the measured currents $I_i^1$ and $I_i^2$ $(i = 1, \cdots, 4)$, respectively, we find the six equations:

$$1 = R_1(I_2^1 + I_3^1 + I_4^1) + I_j^1 R_j, \qquad j = 2, 3, 4 \tag{1.2$_j$}$$

and

$$1 = R_2(I_1^2 + I_3^2 + I_4^2) + I_j^2 R_j, \qquad j = 1, 3, 4. \tag{1.3$_j$}$$

We can easily check that the four equations $(1.2_j)$ and $(1.3_1)$ are linear and independent, thus determining the $R_i$ $(i = 1, \cdots, 4)$.

In this paper we consider the problem of determining the individual resistance values in a fixed network of resistors from direct current voltage and current measurements at certain accessible sites. This is a discrete version of the problem of determining the conductivity ((resistivity)$^{-1}$) of a bounded region in $\mathbb{R}^n$ from steady state direct current measurements at the boundary proposed by Calderon in [C] (see [KVI], [KVII], [KVIII], and [SU1], [SUII], and [SUIII] for some results). Indeed, we follow this approach here and use it to prove a local uniqueness result for certain networks (i.e., small deviations

* Received by the editors October 26, 1987; accepted for publication (in revised form) September 22, 1988.

† Department of Mathematics, Duke University, Durham, North Carolina 27706.

‡ The research of this author was supported by National Science Foundation grant DMS-8502293 and an Alfred P. Sloan Research Fellowship.

§ The research of this author was supported by National Science Foundation grant DMS-8600797.

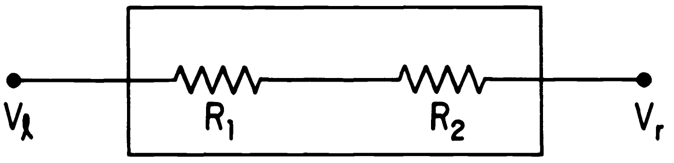


FIG. 1

from a known configuration can be distinguished by experiment). We outline the approach below.

If we fix the voltages at all boundary points (henceforth we shall refer to accessible sites as boundary points), then all the voltages are uniquely determined. If we let $f$ denote the voltages at the boundary, then the interior voltages $F$ solve the Dirichlet problem

$$L_\gamma F(x) := \sum_{y \sim x} \gamma(x,y)(F(y)-F(x)) = 0, \qquad x \in V, \tag{1.4}$$

$$F(y) = f(y); \qquad y \in \partial V \tag{1.5}$$

where we represent our network as a finite simple graph with vertices $\bar{V} = V \cup \partial V$ with at least one boundary point ($y \in \partial V$) in each connected component; $x \sim y$ means that $x$ and $y$ are vertices connected by the edge (resistor) $\{x, y\}$ with conductance $\gamma(x, y)$: $Q_\gamma(f)$ represents the total power necessary to maintain the voltage potential $f$ on $\partial V$ and is given by

$$Q_\gamma(f) := \sum_{\{x,y\}} \gamma(x,y)(F(y)-F(x))^2. \tag{1.6}$$

This can be rewritten (see Lemma 2.1), using summation by parts:

$$Q_\gamma(f) = \sum_{x \in \partial V} \sum_{y \sim x} f(x)\gamma(x,y)(F(y)-F(x)) \tag{1.7}$$

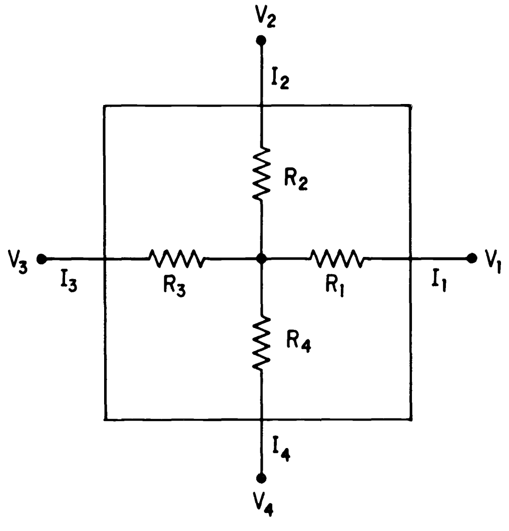


FIG. 2

or

$$Q_\gamma(f) = \sum_{x \in \partial V} \sum_{y \sim x} f(x) I(x,y) \tag{1.8}$$

where $I(x, y) = \gamma(x, y)(F(y) - F(x))$ is the current flow through $\{x, y\}$. From (1.8), we draw the conclusion that $Q_\gamma$ is computable from voltage and current measurements at boundary points. (We assume that we can measure the current flow through each edge which borders a boundary vertex.)

$Q_\gamma$ is a quadratic form; the polarized form is

$$Q_\gamma(f,g) = \sum_{\{x,y\}} \gamma(x,y)(F(x)-F(y))(G(x)-G(y)) \tag{1.9}$$

where $G$ and $g$ are also related by (1.4) and (1.5).

Still following Calderon's approach, we consider the mapping $\Phi$, from conductivities (functions which assign a conductivity to each edge) to quadratic forms, given by

$$\gamma \xrightarrow{\Phi} Q_\gamma(\cdot). \tag{1.10}$$

We differentiate $\Phi$ to obtain the linearized map $D\Phi|_\gamma$ (at $\gamma$). Setting

$$\gamma_\varepsilon = \gamma + \varepsilon\delta\gamma,$$

we have

$$\delta\gamma \xrightarrow{D\Phi|_\gamma} \delta Q = \frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} Q_{\gamma_\varepsilon}.$$

The computation yields (Lemma 2.2)

$$\delta Q(f,g) = \sum_{\{x,y\}} \delta\gamma(x,y)(F(x)-F(y))(G(x)-G(y)). \tag{1.11}$$

Let $\mathbb{R}^E$ denote the vector space of functions on the edges of $\bar{V}$ and let $\mathcal{S}_\gamma$ be the linear span of the functions paired with $\delta\gamma$ in (1.11), that is,

(1.12) $\mathcal{S}_\gamma =$ {linear span of function of the form $(F(x) - F(y))(G(x) - G(y))$ where $F$ and $G$ satisfy (1.4)}.

If $\delta Q(f, g) = 0$ in (1.11), then $\delta\gamma \in \mathcal{S}_\gamma^\perp$ (the set of functions that are $l^2$ orthogonal to $\mathcal{S}_\gamma$); hence $D\varphi|_{\gamma_0}$ is injective if and only if $\mathcal{S}_\gamma^\perp = \{0\}$ (equivalently, $\mathcal{S}_\gamma = \mathbb{R}^E$). Application of the implicit function theorem (injective version) yields the local injectivity of $\Phi$ and therefore Theorem 1.1.

THEOREM 1.1. *If $\mathcal{S}_\gamma = \mathbb{R}^E$, then small deviations from $\gamma$ can be determined from measurements at boundary points.*

Next we restrict to special graphs, namely subsets of the graph whose vertices are the integer lattice $\mathbb{Z}^d$ with nearest neighbor edges ($x \sim y$ if $|x - y| = 1$). Let $E$ be a finite subset of the edges in the lattice and define $\bar{V}$ by

$$y \in \bar{V} \quad \text{if } \{x,y\} \in E \text{ for some } x \tag{1.13}$$

and

$$y \in V \quad \text{if } \{x,y\} \in E \text{ for all } x \text{ with } |x-y| = 1 \tag{1.14}$$

(hence $\partial V = \bar{V} \setminus V$). For such $V$, we show (in § 3) that $\mathcal{S}_1$ ($\mathcal{S}_\gamma$ for $\gamma \equiv 1$) $= \mathbb{R}^E$ and, hence, via Theorem 1.1, we have Theorem 1.2.

THEOREM 1.2. *For $\bar{V}$ defined as above, small deviations from the constant conductivity can be determined from measurements at boundary points.*

For more general networks, the problem of local injectivity reduces to determining when $\mathscr{S}_\gamma = \mathbb{R}^E$. One bound can be given:

$$\dim(\mathscr{S}_\gamma) \leqq \tfrac{1}{2}(|\partial V|)(|\partial V| - 1). \tag{1.15}$$

To see this, for $z \in \partial V$, let $F_z$ be the function satisfying (1.4) with boundary values $F_z(z) = 1$, $F_z(w) = 0$, $w \in \partial V$, $w \neq x$. Then if $f$ and $g$ are any functions satisfying (1.4), we have

$$f(x) = \sum_{z \in \partial V} F_z(x) f(z), \qquad g(x) = \sum_{w \in \partial V} F_w(x) g(w)$$

and

$$(f(x)-f(y))(g(x)-g(y)) = \sum_{z \in \partial V} \sum_{w \in \partial V} f(z) g(w) (F_z(x)-F_z(y))(F_w(x)-F_w(y)).$$

Therefore $\{(\nabla F_z)(\nabla F_w): z, w \in \partial V\}$ spans $\mathscr{S}_\gamma$. Also note that $\nabla(\sum_{w \in \partial V} F_w) = 0$, and hence $(\nabla F_z)(\nabla F_z) = -(\nabla F_z)(\sum_{w \neq z} \nabla F_w)$; therefore $\{(\nabla F_z)(\nabla F_w): z, w \in \partial V, z \neq w\}$ spans $\mathscr{S}_\gamma$ and this set contains $\frac{1}{2}|\partial V|(|\partial V| - 1)$ functions. These functions are not necessarily independent, however. It is an open, apparently difficult problem to decide when $\mathscr{S}_\gamma = \mathbb{R}^E$ for a given network.

**2. Proofs of lemmas.** Let $\mathscr{G} = (\bar{V}, E)$ be a finite simple graph where the set of vertices $\bar{V}$ is divided into $V$ and $\partial V$, where $x \in V$ are called the interior and $x \in \partial V$ the boundary vertices. We assume that each connected component of $\mathscr{G}$ has at least one boundary point (vertex). Let

$$\gamma : E \to (0, \infty)$$

and for $F: \bar{V} \to \mathbb{R}$, define

$$L_\gamma F(x) = \sum_{x \sim y} \gamma(x,y)(F(y) - F(x))$$

where $y \sim x$ means the edge $\{x, y\} \in E$. For any $f: \partial V \to \mathbb{R}$, the unique solution to the Dirichlet problem

$$L_\gamma F(x) = 0 \quad \text{for } x \in V, \qquad F(x) = f(x) \quad \text{for } x \in \partial V \tag{2.1}$$

is given by

$$F(x) = E_x(f(X_\tau))$$

where $X_t$ is the continuous time Markov chain with symmetric rates $\gamma(x, y) = \gamma(y, x)$ and

$$\tau = \inf\{t \geqq 0 : X_t \in \partial V\}.$$

Recall the bilinear form $Q_\gamma(f, g)$ as defined in (1.9); to see that $Q_\gamma$ is determined by boundary measurements, we need the following lemma.

LEMMA 2.1. *If $F$ satisfies* (2.1), *then*

$$\sum_{\{x,y\} \in E} \gamma(x,y)(F(x)-F(y))(G(x)-G(y)) = \sum_{x \in \partial V} \sum_{y \sim x} \gamma(x,y) G(x)(F(x)-F(y)).$$

*Proof.*

$$\begin{aligned}\sum_{\{x,y\}\in E} &\gamma(x,y)(F(x)-F(y))(G(x)-G(y))\\ &=2\sum_{\{x,y\}\in E}\gamma(x,y)G(x)(F(x)-F(y))\\ &=\sum_{x\in V}G(x)L_\gamma F(x)+\sum_{x\in\partial V}\sum_{y\sim x}\gamma(x,y)G(x)(F(x)-F(y))\end{aligned}$$

but the first term is zero by (2.1) and the lemma follows. □

We next turn to the question of identifying $\gamma$ from $Q_\gamma$. We recall (1.10) that the mapping $\Phi$ applied to $\Phi$ is $Q_\gamma$ so that we may write, for any $f, g \in \mathbb{R}^{\partial V}$

$$\Phi(\gamma)(f,g)=Q_\gamma(f,g).$$

We compute the derivative of the map $\Phi$ at $\gamma$ in the direction $\delta\gamma$ by letting

$$\gamma_\varepsilon=\gamma+\varepsilon\delta\gamma$$

and computing $d/d\varepsilon|_{\varepsilon=0}Q_{\gamma_\varepsilon}$. We have Lemma 2.2.

LEMMA 2.2.

$$\begin{aligned}D\Phi|_\gamma\delta\gamma(f,g)&=\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}Q_{\gamma_\varepsilon}(f,g)\\ &=\sum_{\{x,y\}\in E}\delta\gamma(x,y)(F(x)-F(y))(G(x)-G(y))\end{aligned}$$

*where F and G satisfy* (2.1).

*Proof.*

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}Q_{\gamma_\varepsilon}(f,g)=\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}\sum_{\{x,y\}\in E}(\gamma+\varepsilon\delta\gamma)(x,y)(F_\varepsilon(x)-F_\varepsilon(y))(G_\varepsilon(x)-G_\varepsilon(y))$$

where $F_\varepsilon$, $G_\varepsilon$ solve (2.1) with $\gamma_\varepsilon$ in place of $\gamma$ ($F_0 = F$, $G_0 = G$):

$$\begin{aligned}\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}Q_{\gamma_\varepsilon}(f,g)=&\sum_{\{x,y\}\in E}\delta\gamma(x,y)(F(x)-F(y))(G(x)-G(y))\\ &+\sum_{\{x,y\}\in E}\gamma(x,y)(F(x)-F(y))\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}(G_\varepsilon(x)-G_\varepsilon(y))\\ &+\sum_{\{x,y\}\in E}\gamma(x,y)(G(x)-G(y))\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}(F_\varepsilon(x)-F_\varepsilon(y)).\end{aligned}$$

Now, by Lemma 2.1, the second term can be replaced by

$$\sum_{x\in\partial V}\sum_{y\sim x}\gamma(x,y)(F_0(x)-F_0(y))\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}G_\varepsilon(x)$$

and, for $x\in\partial V$

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}G_\varepsilon(x)=\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0}g(x)=0$$

so that the second term vanishes. The third term vanishes just as the second, and the lemma is proved. □

**3. Subnetworks of the cubic lattice.** In this section we prove Theorem 1.2. As has been pointed out in § 1, this is an immediate consequence of the following proposition.

PROPOSITION 3.1. $\mathcal{S}_1 = \mathbb{R}^E(\mathcal{S}_1^\perp = \{0\})$. *The vector space* $\mathcal{S}_1$ ($\mathcal{S}_\gamma$ *for* $\gamma \equiv 1$) *is the subspace of* $\mathbb{R}^E$ *defined in* (1.12) *and E is a finite subset of the edges* $\mathcal{E}$ *in the lattice* $\mathbb{Z}^d$ ($\mathcal{E} = \{\{x, y\} \mid |x - y| = 1\}$).

It will be convenient to note that $\mathcal{E}$ can be put in one-to-one correspondence with the set $\mathbb{Z}^d \oplus \{1, \cdots, d\}$ via the map that sends

$$\{x, x+e_j\} \mapsto (x, j)$$

where $e_j$ is the $j$th unit coordinate vector. Hence, for functions on $\mathcal{E}$, we have

$$\mathbb{R}^{\mathcal{E}} \cong \bigoplus_{j=1}^{d} \mathbb{R}^{\mathbb{Z}^d} \tag{3.1}$$

(i.e., a function on $\mathcal{E}$ can be regarded as a $d$-dimensional vector-valued function on $\mathbb{Z}^d$). If $F(x)$ is a function on vertices we shall denote by $\nabla F$ the function on edges defined by

$$\nabla F(\{x, x+e_j\}) = F(x+e_j) - F(x).$$

Using (3.1), we may consider $\nabla F$ to be a vector-valued function on $V$, with components

$$(\nabla F(x))_j = F(x+e_j) - F(x), \qquad j = 1, \cdots, d.$$

In this notation,

$$\mathcal{S}_1 = \text{span}\,\{(\nabla F)(\nabla G) \mid F, G \text{ are harmonic (satisfy (1.1) with } \gamma \equiv 1) \text{ on } U\}.$$

We turn now to the proof of Proposition 3.1, which we divide into the following two lemmas.

LEMMA 3.2. *Let* $T = \{V \in \mathbb{R}^d \mid 0 < |v_j| < \pi;\ |v_j| \neq |v_k|, j \neq k\}$; *then*

$$\mathcal{T} := \text{span}\,\{we^{ix\cdot v} \mid v \in T, w \in \mathbb{R}^d\} = \mathbb{R}^E.$$

*Proof.* If $\psi \in \mathbb{R}^E$ we will also write $\psi$ for the compactly supported function on $\psi$: $\mathbb{Z}^d \to \mathbb{R}^d$:

$$[\psi(x)]_j = \begin{cases} \psi(x, x+e_j), & \{x, x+e_j\} \in E, \\ 0 \quad \text{otherwise}. \end{cases}$$

Assume $\mathcal{T} \neq \mathbb{R}^E$. Then since $\mathbb{R}^E$ is a finite-dimensional vector space, there exists a nonzero $\psi \in \mathcal{T}^\perp$ such that $\psi \cdot f = 0$ for every $f \in \mathcal{T}$, i.e.,

$$\sum_{x \in \mathbb{Z}^d} (\psi(x) \cdot w) e^{ix\cdot v} = 0, \qquad w \in \mathbb{R}^d, \quad v \in T. \tag{3.2}$$

The left-hand side of (3.2) is the Fourier transform $\hat{\psi}_w$ of $\psi_w = \psi \cdot w$, which is compactly supported. If $\bar{T}$ is the periodic extension of $T$, i.e., $\bar{T} = \{v + 2\pi(j_i, \cdots, j_d): v \in T, j_i \in \mathbb{Z}\}$, then the Lebesgue measure of $\bar{T}^c$ is zero and $\hat{\psi}_w(v) = 0$ for $v \in \bar{T}$. But since $\psi \cdot w$ is compactly supported, $\hat{\psi}_w$ is continuous. Therefore, $\hat{\psi}_w \equiv 0$ and by uniqueness of the Fourier transform, $\psi \cdot w \equiv 0$. Since this is true for every $w \in \mathbb{R}^d$, $\psi \equiv 0$, which is a contradiction. □

LEMMA 3.3. $\mathcal{S}_1 \supseteq \mathcal{T}$.

*Proof.* We shall show that for any $v \in T$, there exist $d$ complex vectors

$$z^i \in \mathbb{C}^d, \qquad i = 1, \cdots, d$$

such that

$$e^{x\cdot z^i} \text{ and } e^{-x\cdot \bar{z}^i} \text{ are harmonic,} \tag{3.3}$$

$$(\nabla e^{x\cdot z^i})(\nabla e^{-x\cdot \bar{z}^i}) = w^i e^{ix\cdot v}, \tag{3.4}$$

$$w^1, \cdots, w^d \text{ span } \mathbb{C}^d. \tag{3.5}$$

The lemma is then a direct consequence of (3.3)–(3.5).

To establish (3.3)–(3.5), we shall treat separately the cases $d > 2$ and $d = 2$.

($d > 2$). If $z \in \mathbb{C}^d$, then $e^{z\cdot x}$ (and $e^{-\bar{z}\cdot x}$) is harmonic if

$$\sum_{j=1}^{d} \cosh z_j = d. \tag{3.6}$$

If $v \in T$, and $z_j = a_j + i(v_j/2)$, (3.6) becomes

$$f_1(\alpha) = \sum_{j=1}^{d} \alpha_j \beta_j = 0, \tag{3.7}$$

$$f_2(\alpha) = \sum_{j=1}^{d} (1+\alpha_j^2)^{1/2}(1-\beta_j^2)^{1/2} - d = 0 \tag{3.8}$$

where

$$\alpha_j = \sinh a_j, \qquad \beta_j = \sin \frac{v_j}{2}$$

(we have used here $\cos v_j/2 > 0$).

Fix $v$, and hence $\beta$, and let $\eta = (\eta_1, \cdots, \eta_d)$ be any unit vector with $\sum_{j=1}^{d} \eta_j \beta_j = 0$. If we let $\alpha = r\eta$ where $r > 0$, then

$$f_2(\alpha) = G(r) - d$$

where

$$G(r) = \sum_{j=1}^{d} (1 + r^2\eta_j^2)^{1/2}(1-\beta_j^2)^{1/2}. \tag{3.9}$$

Because $G(r)$ increases to infinity and $G(0) < d$, there is a unique positive $r_0$ with $G(r_0) = d$. Let $\alpha = r_0\eta$ so that $\alpha$ satisfies (3.7) and (3.8) and if we let $z = (z_1, \cdots, z_d)$ with

$$z_j = \sinh^{-1}(\alpha_j) + i\frac{v_j}{2}, \tag{3.10}$$

then $z$ satisfies (3.6). A computation yields that

$$w_j(\alpha_j) = (\nabla e^{z\cdot x})_j (\nabla e^{-\bar{z}\cdot x})_j e^{-iv\cdot x} = e^{iv_j/2}((1+\alpha_j^2)^{1/2} - (1-\beta_j^2)^{1/2}). \tag{3.11}$$

Hence,

$$w(\alpha) = (w_1, \cdots, w_j) = \operatorname{diag}(e^{iv_1/2} \cdots e^{iv_d/2})\tilde{w}(\alpha) \tag{3.12}$$

where

$$\tilde{w}_j(\alpha) = (1+\alpha_j^2)^{1/2} - (1-\beta_j^2)^{1/2}.$$

We claim that only the zero vector is orthogonal to $\tilde{w}(\alpha)$ for every $\alpha$ satisfying (3.6) and (3.7). According to (3.12) this will imply the same conclusion for $w(\alpha)$, and hence we may choose $d$ vectors $\alpha^1, \cdots, \alpha^d$ such that $w(\alpha^1), \cdots, w(\alpha^d)$ span $\mathbb{C}^d$. If we define $z^1, \cdots, z^d$ by (3.10) we will then satisfy (3.3)–(3.5).

To establish the claim, suppose that there exists a $v \in \mathbb{C}^d$ such that for every $\alpha$ satisfying (3.6) and (3.7), we have

$$f_3(\alpha) = \tilde{w}(\alpha)\cdot v = 0. \tag{3.13}$$

Now the components of $\tilde{w}(\alpha)$ are positive so $v$ must have both positive and negative components. Say that $v_1 > 0$ and $v_2 < 0$. Let $\tilde{\alpha} = (\alpha_1, \alpha_2, 0, \cdots, 0)$ be the unique solution to (3.7) and (3.8) with $\alpha_j = 0$ for $j \geqq 3$. (We can find such a solution by starting with the unit vector $\eta = (\beta_1^2 + \beta_2^2)^{-1/2}(\beta_1, \beta_2, 0, \cdots, 0)$.) If we differentiate at $\tilde{\alpha}$ we obtain

$$df_1(\tilde{\alpha}) = \sum_{j=1}^{d} \beta_j d\alpha_j, \qquad \beta_j \neq 0, \quad j = 1, \cdots, d,$$

$$df_2(\tilde{\alpha}) = \frac{\alpha_1}{(1+\alpha_1^2)^{1/2}}(1-\beta_1^2)^{1/2} d\alpha_1 + \frac{\alpha_2}{(1+\alpha_2^2)^{1/2}}(1-\beta_2^2)^{1/2} d\alpha_2,$$

$$df_3(\tilde{\alpha}) = \frac{\alpha_1 v_1}{(1+\alpha_1^2)^{1/2}} d\alpha_1 + \frac{\alpha_2 v_2}{(1+\alpha_2^2)^{1/2}} d\alpha_2$$

and note that $df_1(\tilde{\alpha})$, $df_2(\tilde{\alpha})$, and $df_3(\tilde{\alpha})$ are linearly independent (use $v_1 > 0$ and $v_2 < 0$). Therefore, we may perturb $\tilde{\alpha}$, remaining on the manifold $f_1(\alpha) = f_2(\alpha) = 0$, and changing $f_3$ to be nonzero, contradicting (3.13).

($d = 2$). Let $v = (v_1, v_2)$ and assume that $0 < |v_1| < |v_2| < \pi$ ($|v_1| > |v_2|$ is similar) and hence that

$$0 < \cos\frac{v_2}{2} < \cos\frac{v_1}{2} < 1, \tag{3.6}$$

$$0 < \sin^2\frac{v_1}{2} < \sin^2\frac{v_2}{2} < 1. \tag{3.7}$$

Since

$$0 < \cos\frac{v_1}{2} \pm \cos\frac{v_2}{2} < 2,$$

(3.6), (3.7), and the intermediate value theorem imply that there exist unique positive solutions $r_+$ and $r_-$ to

$$\left(1 + r_\pm^2 \sin^2\frac{v_2}{2}\right)^{1/2} \cos\frac{v_1}{2} \pm \left(1 \pm r_\pm^2 \sin^2\frac{v_1}{2}\right)^{1/2} \cos\frac{v_2}{2} = 2. \tag{3.8$_\pm$}$$

If we set

$$z^1 = \left(\sinh^{-1}\left(r_+ \sin\frac{v_1}{2}\right) \sinh^{-1}\left(-r_+ \frac{\sin v_2}{2}\right)\right) + i\left(\frac{v_1}{2}, \frac{v_2}{2}\right)$$

and

$$z^2 = \left(\sinh^{-1}\left(r_- \sin\frac{v_1}{2}\right), \sinh^{-1}\left(-r_- \sin\frac{v_2}{2}\right)\right) + i\left(\frac{v_1}{2}, \frac{v_2}{2} + \pi\right),$$

(3.3) and (3.4) can be checked directly, and the computation gives

$$w_j^i = (e^{z_j^i} - 1)(e^{-\bar{z}_j^i} - 1)$$

and

$$\det\{w^1, w^2\} = 2\exp\left(i\left(\frac{v_1+v_2}{2}\right)\right)\left(\cos\frac{v_1}{2}\right)^{-1}\left(\cos^2\frac{v_1}{2}+\cos^2\frac{v_2}{2}\right)$$
$$\cdot\left(\left(1+r_+^2\sin^2\frac{v_1}{2}\right)^{1/2}+\left(1+r_+^2\sin^2\frac{v_2}{2}\right)^{1/2}\right),$$

which cannot be zero. Thus (3.5) follows. □

## REFERENCES

[C] A. P. CALDERON, *On an inverse boundary value problem*, in Seminar on Numerical Analysis and Its Applications to Continuum Physics, Soc. Brasileira de Matematica, Rio de Janerio, 1980, pp. 65–73.

[KVI] R. KOHN AND M. VOGELIUS, *Determining conductivity by boundary measurements*, Comm. Pure Appl. Math., 37 (1984), pp. 113–123.

[KVII] ———, *Determining conductivity by boundary measurements* II. *Interior results*, Comm. Pure Appl. Math., 38 (1985), pp. 643–667.

[KVIII] ———, *Identification of an unknown conductivity by means of measurements at the boundary*, Proc. SIAM–AMS Symposium on Inverse Problems, New York, 1983.

[SUI] J. SYLVESTER AND G. UHLMANN, *A uniqueness theorem for an inverse boundary value problem in electrical prospection*, Comm. Pure Appl. Math., 39 (1986), pp. 91–112.

[SUII] ———, *A global uniqueness theorem for an inverse boundary value problem*, Ann. of Math., 125 (1987), pp. 153–169.

[SUIII] ———, *Inverse boundary value problems at the boundary-continuous dependence*, Comm. Pure Appl. Math., 41 (1988), pp. 197–219.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 240–244, May 1989

# A ROBUST NONCRYPTOGRAPHIC PROTOCOL FOR COLLECTIVE COIN FLIPPING*

MICHAEL SAKS†

**Abstract.** A new protocol for global coin flipping in the model of Ben-Or and Linial is presented. In this model, global coin flipping is considered to be an asynchronous perfect information game among $n$ players, some of whom are dishonest. Each player possesses a fair coin and they wish to agree on the value of a single bit, which the honest players want to be random. A protocol is $\varepsilon$-robust for $t$ dishonest players if no set of $t$ dishonest players can bias the bit by more than $\varepsilon$. The protocol given here is $\varepsilon$-robust against $\theta(n/\log n)$ dishonest players for any fixed $\varepsilon$, improving on the $\theta(n^{.63\cdots})$ robust protocol given by Ben-Or and Linial.

**Key words.** collective coin flipping, perfect information games

**AMS(MOS) subject classifications.** 68R05, 90D99, 05A05

## 1. Introduction.

A set of $n$ players wish to agree on a single bit value (0 or 1) which they will all accept as random. Each player possesses a fair coin that he flips privately. One obvious way to do this is to have one of the players flip his coin and announce the outcome. If, however, the designated player is dishonest, he may announce whatever bit value he chooses. The problem of *collective coin flipping* is to design a procedure for $n$ players to agree on a bit which is robust in the sense that even in the presence of dishonest players who conspire to bias the bit, the outcome is unbiased or nearly unbiased.

The precise formulation of the coin flipping problem depends on assumptions about how the players communicate and the computational power of the players (see [BLS] for a survey). In this paper, we consider a model first formalized by Ben-Or and Linial [BL], in which collective coin flipping is an *asynchronous perfect information game*. In such a game, the players take turns announcing bits which are supposed to be obtained from fair coins, and the output bit is the value of some prespecified function of the bits. (We will give a more precise definition of these games in § 2.) A simple example is the *majority game* in which each player supplies a bit, and the outcome is the majority value. Players who are dishonest may try to influence the outcome by supplying nonrandom bits. We assume that the set of dishonest players may collude to achieve their desired outcome; thus, it is convenient to view them as being under the direction of a single *adversary*, whose aim is to bias the bit in one direction or the other.

A given collective coin flipping game is said to be *$\varepsilon$-robust* against $t$ cheaters if no subset of $t$ cheaters can bias the outcome by more than $\varepsilon$. For instance, the asymptotic behavior of the binomial coefficients implies that for any positive $\varepsilon$ the majority game on $n$ players is $\varepsilon$-robust against $c(\varepsilon)n^{1/2}$ cheaters for some positive constant $c(\varepsilon)$. A basic question about collective coin flipping games is: What is the maximum number $t(n, \varepsilon)$ such that there exists a collective coin flipping game that is *$\varepsilon$-robust* against $t(n, \varepsilon)$ cheaters? The majority game shows that for any $\varepsilon$, $t(n, \varepsilon) = \Omega(n^{1/2})$. Ben-Or and Linial, who posed the question, described a game that is *$\varepsilon$-robust* against $\Omega(n^{.63\cdots})$ cheaters. In this note, a game that is *$\varepsilon$-robust* against $\Omega(n/\log n)$ cheaters is described, proving the following theorem.

THEOREM 1.1. $t(n, \varepsilon) = \Omega(n/\log n)$.

There are no known nontrivial upper bounds on $t(n, \varepsilon)$. An elementary argument shows that $t(n, \varepsilon) < n/2$ for any $\varepsilon < 1$, but, for instance, it is not known whether

* Received by the editors March 24, 1988; accepted for publication (in revised form) September 7, 1988.

† Department of Mathematics and RUTCOR, Rutgers University, New Brunswick, New Jersey 08903 and Bellcore, Morristown, New Jersey 07960. This work was supported in part by National Science Foundation grant DMS87 03541 and Air Force Office of Scientific Research grant AFOSR-0271.

$t(n, \varepsilon)$ is sublinear in $n$ (or even whether $t(n, \varepsilon) \leqq n/3$). Recently, Kahn, Kalai, and Linial [KKL] have found a beautiful proof that, for collective coin flipping schemes in which each participant supplies exactly one bit, no scheme can be $\varepsilon$-robust against more than $O(n/\log n)$ cheaters. The scheme described in this paper is not of this restricted form; the best-known scheme of this form is the above-mentioned scheme in [BL] that is robust for $\Omega(n^{.63\cdots})$ cheaters.

The original motivation for this problem was the problem of Byzantine Agreement ([Ra]; see the survey by Chor and Dwork [CD] for a survey and references). In this context, players represent communicating processors that, for some distributed computation, need to generate random bits, and these bits should be random (or nearly random) even if some of the processors are faulty (these correspond to dishonest players). Most of the papers dealing with this problem use cryptographic techniques to perform the global coin flip. Such techniques are precluded in the model studied here, because the computation power of the participants is unrestricted.

**2. Definitions.** A *collective coin flipping game* **G** on player set $\{1, 2, \cdots, n\}$ is a rooted tree $T$ with labels on the nodes as follows. Each internal node $v$ is labeled by some player $P_v$ (who is said to *own* that node) and by a discrete probability distribution $D_v$ on its children. Each leaf is labeled by either zero or one. The game is played by starting at the root and traveling down to a leaf. At each internal node, the owner announces which branch to take. The outcome of the game is the label of the leaf reached.

For a game **G**, and $i \in \{0, 1\}$, we define $p_i^{\mathbf{G}}$ to be the probability that the outcome of the game is $i$ if at each internal node $v$ the owner selects the branch according to the distribution $D_v$. The game **G** is *unbiased* if $p_1^{\mathbf{G}} = p_0^{\mathbf{G}} = \frac{1}{2}$.

To play such a game, each player must be able to generate random variables according to the distributions $D_v$. In the introduction, we said only that each player possesses a fair coin. However, it is easy to see that using a fair coin, any discrete distribution can be approximated to any desired accuracy. Furthermore, it is well known that any discrete distribution can be generated exactly in bounded expected time using a fair coin (for instance, to generate the uniform distribution on three items, associate them to 01, 10, and 11, and generate two random bits. If one of these outcomes occurs, then the corresponding item is selected; if 00 appears, then repeat). For more details, see [vN]. Thus, we will assume that players can generate arbitrary discrete distributions.

Now let us consider the effect on the outcome of the game if some subset $S$ of the players is dishonest. At each internal node owned by a dishonest player, the branch taken is chosen by the player. Thus a strategy for the set $S$ of players specifies, for each internal node owned by a member of that set, which branch is to be selected. For the outcome $i$, we define $p_i^{\mathbf{G}}(S)$ to be the maximum over all strategies for $S$ of the probability that the outcome is $i$.

As an example, consider the majority game described in the introduction. For any set of dishonest players, if they want to bias the outcome to 1 (respectively, 0) then they should all announce 1 (respectively, 0) at their turn, and for $S$ of cardinality $s$, $p_i^{\mathbf{G}}(S)$ is equal to the probability that, out of $n - s$ random bits, at least $n/2 - s$ are equal to $i$.

A game **G** is said to be *$\varepsilon$-robust against $t$ cheaters* if, for any set $S$ of at most $t$ players, and $i \in \{0, 1\}$, $p_i^{\mathbf{G}}(S) \leqq \frac{1}{2} + \varepsilon$, i.e., no set of more than $t$ cheaters can bias the outcome by more than $\varepsilon$ away from $\frac{1}{2}$ in either direction. The function $t(n, \varepsilon)$ is defined to be the maximum number $t$ such that there exists a game which is $\varepsilon$-robust against $t(n, \varepsilon)$ cheaters.

The game described in the next section, which achieves the result of Theorem 1.1, is based on the idea of a different kind of game, called an *election game*. An election game **E** is defined similarly to a coin flipping game, except the outcome (i.e., each leaf label) is a number from 1 to $n$, identifying a player, who is said to be elected. For an election game $\mathbf{E}$, the *influence* $I^{\mathbf{E}}(S)$ of a set of dishonest players is the maximum probability over all strategies for $S$ that a member of $S$ is elected.

Associated to any election game $\mathbf{E}$ is a coin flipping game $\mathbf{G}(\mathbf{E})$, obtained by having the elected player flip his coin. It is easy to see that $p_l^{\mathbf{G}(\mathbf{E})}(S) = (1 + I^{\mathbf{E}}(S))/2$, for all sets $S$. Hence, if $\mathbf{E}$ is an election game with $I^{\mathbf{E}}(S) \leqq 2\varepsilon$ for any set of size $t$, then $\mathbf{G}(\mathbf{E})$ is $\varepsilon$-robust against $t$ cheaters.

**3. Pass the baton: A robust leader election game.** In this section we present a simple $n$-player leader election game that is $\varepsilon$-robust against $\theta(n/\log n)$ cheaters, for any $\varepsilon$. The game, called **pass the baton,** works as follows. Initially the baton is held by Player 1. Player 1 randomly selects one of the other players and gives that player the baton. The selected player then gives the baton to a randomly selected player from among the players who have not yet had the baton. This is repeated until each player has held the baton. The leader is the last player to receive the baton.

It is clear that if all players are honest, then each player, except the starting player, is elected with probability $1/(n-1)$. The relevant question is, how many cheaters can there be so that the probability that a cheater is elected remains small?

THEOREM 3.1. *For any $\varepsilon > 0$, there exist constants $c(\varepsilon)$ and $d(\varepsilon)$ such that the algorithm pass the baton is $\varepsilon$-robust against $c(\varepsilon)n/\log n$ cheaters, but not against $d(\varepsilon)n/\log n$ cheaters.*

*Proof.* For purposes of analysis all players, except for the initial player, are equivalent. Thus given the right to select $t$ dishonest players, the only choices that the adversary has are (i) whether the set of dishonest players includes the initial player, and (ii) whenever a dishonest player holds the baton, does he give it to an honest or dishonest player? Intuitively, since the adversary wants to maximize the chance that the last player selected is dishonest, the obvious strategy is not to include Player 1 in the set of cheaters and always have a cheater pass the baton to an honest player. This can be established formally as follows. Let $f(s, t)$ denote the probability that with $s$ unselected honest players and $t$ unselected dishonest players a dishonest player is elected (under the optimal strategy by the dishonest players) given that the baton is currently owned by an honest player. Let $g(s, t)$ denote the corresponding probability given that the baton is currently owned by a dishonest player. Then since a dishonest player can decide who next holds the baton, while an honest player chooses the next player uniformly at random, we have the following recurrences for $f(s, t)$ and $g(s, t)$:
For $s, t \geqq 1$,

$$g(s,t) = \max\{f(s-1,t), g(s,t-1)\} \tag{3.1}$$

$$f(s,t) = \frac{s}{s+t}f(s-1,t) + \frac{t}{s+t}g(s,t-1), \tag{3.2}$$

with the initial conditions $f(0, 0) = 0$, $g(0, 0) = 1$, $g(s, 0) = f(s, 0) = 0$ for all $s \geqq 1$, and $g(0, t) = f(0, t) = 1$ for all $t \geqq 1$.

To establish that the optimal strategy of the adversary is as claimed, it suffices to show that $f(s, t+1) \geqq g(s+1, t)$ since then, for a fixed number of players and dishonest players, it is preferable for the adversary that the baton holder be honest. This follows by induction (on $s$ and $t$) using the following chain of relations:

$$f(s,t+1) = \frac{s}{s+t+1}f(s-1,t+1) + \frac{t+1}{s+t+1}g(s,t) \geqq g(s,t) \geqq f(s,t) = g(s+1,t).$$

The first equality is (3.2). The induction hypothesis implies $f(s-1, t+1) \geqq g(s, t)$, which implies the first inequality. The next inequality follows from (3.1) and (3.2), since $g(s, t)$ is the maximum of two quantities, while $f(s, t)$ is a convex combination of the same two quantities. Finally, by the induction hypothesis, $f(s, t) \geqq g(s+1, t-1)$; thus, by (3.1), $g(s+1, t) = f(s, t)$.

Hence, (3.1) reduces to $g(s, t) = f(s-1, t)$ and (3.2) becomes

$$f(s,t)=\frac{s}{s+t}f(s-1,t)+\frac{t}{s+t}f(s-1,t-1), \tag{3.3}$$

for $s, t \geqq 1$.

It is useful to think of the behavior of the algorithm in terms of a procedure for removing balls from an urn. Consider an urn containing $s$ white balls and $t$ red balls. Remove balls from the urn according to the following rules: select a ball uniformly at random. After that, if the last selected ball is a white ball, then select the next ball at random from the urn. If the last selected ball is red, then the next ball to be selected is a white ball (unless no white balls remain).

The recurrence (3.3) does not appear to have a closed form solution but the following bounds do hold.

LEMMA 3.2. *For $t \geqq 1$ and $s \geqq 0$,*

$$1-2\ln 2\left(\frac{s}{(t+1)\ln(t+1)}\right)\leqq f(s,t)\leqq\frac{t\ln(t+1)}{(s+1)\ln 2}. \tag{3.4}$$

From this lemma, it is easy to conclude that for $t < \varepsilon s/\log_2 s$, $f(s, t) < \varepsilon$ and for $s < \varepsilon(t+1)\ln(t+1)/2\ln 2$, $f(s, t) > 1-\varepsilon$, from which the theorem follows.

*Proof of the Lemma.* The upper bound is proved first using induction on both $s$ and $t$. The basis step $t = 1$ or $s = 0$ are immediate. The induction step has two cases.

*Case* i. $s \leqq \ln t/(\ln(t+1) - \ln t)$. Then the right-hand side is greater than or equal to $t \log_2(1 + 1/t)$, which is at least 1 for $t \geqq 1$, and the inequality holds trivially.

*Case* ii. $s > \ln t/(\ln(t+1) - \ln t)$. Applying (3.3) and the induction hypothesis yields:

$$\begin{aligned} f(s,t)&\leqq\frac{s}{s+t}\left(\frac{t\ln(t+1)}{s\ln 2}\right)+\frac{t}{s+t}\left(\frac{(t-1)\ln t}{s\ln 2}\right),\\ &=\frac{t}{(s+t)\ln 2}\left(\ln(t+1)+\frac{(t-1)\ln t}{s}\right). \end{aligned} \tag{3.5}$$

We need to show that this is less than the right-hand side of (3.4), i.e.,

$$\frac{1}{s+t}\left(\ln(t+1)+\frac{(t-1)\ln t}{s}\right)\leqq\frac{\ln(t+1)}{s+1},$$

which is equivalent to the case assumption.

The lower bound is also proved by induction on $s$ and $t$; again the basis, $s = 0$ or $t = 1$, is trivial. If the term on the left-hand side of (3.4) is negative, the inequality is immediate, so assume that it is positive, i.e.,

$$2s\ln 2\leqq(t+1)\ln(t+1). \tag{3.6}$$

Applying the induction hypothesis to (3.3) yields:

$$1-f(s,t)\leqq\frac{(s-1)2\ln 2}{(s+t)}\left(\frac{s}{(t+1)\ln(t+1)}+\frac{1}{\ln t}\right).$$

We want to show that the right-hand side is less than or equal to $(2 \ln 2)s/(t+1)\ln(t+1)$. An elementary calculation shows that this is equivalent to $\ln(t+1)/(\ln(1+1/t) \geqq s$, which follows (with a little calculus) from (3.6).

**Acknowledgments.** I would like to thank Ira Gessel, Nathan Linial, Avi Wigderson, and Edna Wigderson for several useful discussions, and Andres Fundia for commenting on an earlier version of the paper.

## REFERENCES

[BL] M. Ben-Or and N. Linial, *Collective coin flipping*, Advances in Computing Research, Silvio Micali, ed., to appear. (Preliminary version: *Collective coin flipping, robust voting schemes and minimal Banzhaf values*, Proc. 26th IEEE Symposium on Foundations of Computer Science, 1985, pp. 408–416.)

[BLS] M. Ben-Or, N. Linial, and M. Saks, *Collective coin flipping and other models of imperfect randomness*, Proceedings of 7th Hungarian Conference on Combinatorics, Colloquia Mathematica Societatis János, Bolyai, 1987, pp. 77–112.

[BD] A. Z. Broder and D. Dolev, *Flipping coins in many pockets (Byzantine agreement on uniformly random values)*, Proc. 25th IEEE Symposium on Foundations of Computer Science, 1985, pp. 408–416.

[CD] B. Chor and C. Dwork, *Randomization in Byzantine agreement*, Advances in Computing Research, Silvio Micali, ed., to appear.

[KKL] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on Boolean functions*, Proc. 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 68–80.

[Ra] M. O. Rabin, *Randomized Byzantine generals*, Proc. 24th IEEE Symposium on Foundations of Computer Science, 1983, pp. 403–409.

[vN] L. von Neumann (written by George E. Forsythe), *Various techniques used in connection with random digits*, J. Res. Nat. Bur. Stand. Appl. Math. Series 12 (1951), pp. 36–38. (Also found in John von Neumann: Collected Works, Vol. 5, Pergamon Press (1963), pp. 768–770.)

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 245-254, May 1989

# GEOMETRIC CONTAINMENT AND PARTIAL ORDERS*

NICOLA SANTORO†, JEFFREY B. SIDNEY‡, STUART J. SIDNEY§, AND JORGE URRUTIA¶

**Abstract.** Given two geometric sets $A$ and $B$, it is said that $A$ is *containable* in $B$ provided $A$ is isometric to a subset of $B$. Containability induces a partial order on any set of geometric figures, such as rectangles in the plane. A recent result states that for the set of rectangles in the plane, the containability partial order is of countably infinite dimension. In this paper the rectangle result is extended to other families of geometric figures and to a partial order obtained from quadratic polynomials.

**Key words.** partial order, rectangle containment, posset dimension

**AMS(MOS) subject classifications.** primary 06A10; secondary 06B05, 05A05

**1. Introduction.** In recent years, the study of relationships between geometry and partial orders has attracted the attention and the interest of many researchers from different fields, as witnessed by the large number of results on the subject published in the last few years (for a survey, see [16]). Depending on whether the geometric objects under consideration are fixed in the plane (the "static" case) or can be moved in the plane through translation, rotation, or even reflection (the "dynamic" case), different problems arise and have been studied. The majority of the investigations in both cases have focused on "simple" geometric figures such as rectangles [1], [7], [11], [12], [16], polygons [3], [13], [15], [16], circles [2], [14]-[16], and angular regions [5]-[7], [13], [16].

In this paper, we continue the investigation of the dynamic case started in [12]. Our focus will be on the *containability* relation. Given geometric sets $A$ and $B$, we say that $A$ is *containable* in $B$ provided $A$ is isometric to a subset of $B$; in this case we write $A \sqsubseteq B$. Note that containability defined on a set of objects defines a partial order called the *containability* partial order.

The following question will be of interest:

Given a set of geometric figures, what is the dimension of the containability partial order? In particular, when is the dimension finite?

In the case of finite dimension, containability can be reduced to *finite-dimensional* vector *dominance* using the standard product order $\leqq$ for $\mathscr{R}^k$: $x = (x_1, \cdots, x_k) \leqq y = (y_1, \cdots, y_k)$ if and only if $x_i \leqq y_i$ for $1 \leqq i \leqq k$. The problem of determining whether a figure is containable in another is of principal interest in computational geometry; thus an answer to the above questions would be of immediate practical relevance due to the existence of efficient computational methods for determining dominance relationships among vectors [8], [9]. Furthermore, reduction to vector dominance has already been successfully employed to solve other basic geometric problems [4], [10], [17]. Also, it is not hard to envision applications of positive results to packing problems as well as others.

For some families $H$ of figures, this reduction can be easily accomplished. For example, $f(A)$ defined as the area of $A$ will work for the family $P_k$ of regular polygons with

* Received by the editors December 17, 1987; accepted for publication (in revised form) October 24, 1988. This research was supported under Natural Sciences and Engineering Research Council of Canada grants A2415, A2507, and A0977.

† School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6.
‡ Faculty of Administration, University of Ottawa, Ottawa, Ontario, Canada K1N 9B4.
§ Department of Mathematics, University of Connecticut, Storrs, Connecticut 06268.
¶ Department of Computer Science, University of Ottawa, Ottawa, Ontario, Canada K1N 9B4.

$k \geqq 3$ sides, as well as for the family $C$ of circles. A more interesting example is the family $E$ of ellipses: for each $E_i \in E$, define $f(E_i) = (x_i, y_i)$ where $x_i$ and $y_i$ denote the length of the minor and major axis of $E_i$, respectively; it is easy to show that $E_i$ is containable in $E_k$ if and only if $f(E_i) \leqq f(E_k)$. Similarly, two parameters (namely, the lengths of the diagonals) suffice for rhombi.

In the opposite direction, it has been shown that finitely many parameters do not suffice for plane rectangles [12]; this result also implies that a finite reduction does not exist for convex polygons with at least $k \geqq 4$ sides. On the other hand, in the same paper it is also shown that a reduction is possible using a countable number of parameters.

Many families of figures commonly considered do have "natural" finite parameterizations, but these may not faithfully reflect the containability relation. Typically these parameterizations do possess natural monotonicity and homogeneity properties (example: length and width for rectangles).

In § 2 parameterizations with these properties (denoted (M) and (H)) are studied and used to prove an abstract version of the rectangle theorem, namely that a certain partial order has nonfinite dimension. In § 3, it is shown how the proof of the infinite dimensionality of rectangular containability [12] can be formulated as an instance of the abstract theorem; and these results are extended to the classes of right circular cylinders and of isosceles triangles. It is also shown how low-dimensional results (e.g., for rectangles) can be extended to higher dimensions (e.g., rectangular solids). In § 4 the abstract theorem is used to show that a certain natural algebraic partially ordered set cannot be faithfully represented by finitely many parameters. Finally, § 5 echoes § 4 of [12], displaying a representation of the family of (congruence classes of) nonempty compact sets in $\mathscr{R}^k$ by countably many parameters that are continuous in an appropriate sense. From this result, it follows that the dimension of several previously discussed partial orders are *countably* infinite.

**2. Preliminaries and the abstract theorem.** Let $\underline{P} = (P, \sqsubseteq)$ be any partially ordered set (or poset). Thus $P$ is a nonempty set and $\sqsubseteq$ is a transitive binary relation on $P$ such that for elements $a$, $b$ of $P$, $a \sqsubseteq b$ and $b \sqsubseteq a$ if and only if $a = b$. Any injection $f : P \twoheadrightarrow \mathscr{R}^I$ induces a partial order $\tau_f$ on $f(P) = \{f(a) : a \in P\}$ by the rule $f(a)\tau_f f(b)$ if and only if $a \sqsubseteq b$. If $\tau_f$ coincides with the restriction of $\leqq$ to $f(P)$, we say that $f$ reduces $\sqsubseteq$ to vector dominance in $\mathscr{R}^I$. It is easy to see that any partial order can be reduced to vector dominance in $\mathscr{R}^I$ for some (possibly infinite) index set $I$. Indeed, if $\underline{P} = (P, \sqsubseteq)$ is any poset, take $I = P$ and define $f : P \twoheadrightarrow \{0, 1\}^I \subseteq \mathscr{R}^I$ by $f(a)_i = 0$ if $a \sqsubseteq i$ and $f(a)_i = 1$ otherwise; clearly $a \sqsubseteq b$ implies $f(a) \leqq f(b)$, while if $a \sqsubseteq b$ is false, then $f(a)_b = 1$ while $f(b)_b = 0$, so $f(a) \leqq f(b)$ is false. Note that the same construction of $f$ will work if instead of $I = P$ we take $I = S$ for any subset $S$ of $P$ which is separating in the sense that $a \neq b$ implies $\{i \in S : a \sqsubseteq i\} \neq \{i \in S : b \sqsubseteq i\}$; this observation will be of some use later.

We now state the abstract theorem. Let $\mathscr{R}^k_+ = \{x = (x_1, \cdots, x_k) \in \mathscr{R}^k : x_i > 0, 1 \leqq i \leqq k\}$. A subset $K$ of $\mathscr{R}^n$ is a cone if $K \neq \varnothing$ and $tx \in K$ whenever $x \in K$ and $t$ is a positive (greater than zero) real number.

THEOREM 1. *Suppose that $K$ is a cone in $\mathscr{R}^k_+$ and $\sqsubseteq$ is a partial order on $K$ that satisfies the monotonicity* (M), *homogeneity* (H), *and convergence* (C) *properties below*:

(M) *For all $x, y \in K$, if $x \leqq y$ then $x \sqsubseteq y$.*

(H) *For all $x, y \in K$, if $t > 0$ and $x \sqsubseteq y$ then $tx \sqsubseteq ty$.*

(C) *There exist distinct points $z, w \in K$ and sequences $\{x^{(n)}\}$, $\{y^{(n)}\}$ in $K$ such that*

$$\text{For all } n, x^{(n)} \sqsubseteq w \text{ and } z \sqsubseteq y^{(n)} \sqsubseteq w; \tag{2.1}$$

(2.2) *For all $n$ it is false that $x^{(n)} \sqsubseteq y^{(n)}$; and*

(2.3) $x^{(n)} \to z$ *and* $y^{(n)} \to w$ *as* $n \to \infty$.

*Then* dim $(K, \sqsubseteq)$ *is not finite.*

*Proof.* Suppose, to obtain a contradiction, that $f = (f_1, \cdots, f_m) : K \to \mathscr{R}^m$ reduced $\sqsubseteq$ to vector dominance in $\mathscr{R}^m$ for some finite $m$. Each of the $2m$ functions $t \to f_i(tz)$, $t \to f_i(tw)$ is nondecreasing for $0 < t < \infty$ by (M), so they have a common point of continuity $t_0$. By (H) we may replace $x^{(n)}$, $y^{(n)}$, $z$, $w$ by $t_0x^{(n)}$, $t_0y^{(n)}$, $t_0z$, $t_0w$, respectively, without affecting the hypotheses, so we may suppose $t_0 = 1$. Given a positive number $\varepsilon$, there is a positive number $\delta < 1$ such that $|t - 1| \leqq \delta$ implies $|f_i(tz) - f_i(z)| < \varepsilon$ and $|f_i(tw) - f_i(w)| < \varepsilon$ for all $i$. Thus by (M)

$$U = \{x \in \mathscr{R}^k : (1-\delta)z \leqq x \leqq (1+\delta)z\}$$

and

$$V = \{y \in \mathscr{R}^K : (1-\delta)w \leqq y \leqq (1+\delta)w\}$$

are neighborhoods of $z$ and $w$, respectively, in $\mathscr{R}^k$ such that $x \in U \cap K$ implies $|f_i(x) - f_i(z)| < \varepsilon$ for all $i$, and $y \in V \cap K$ implies $|f_i(y) - f_i(w)| < \varepsilon$ for all $i$. In other words, $z$ and $w$ are points of continuity of each $f_i$ as a function of a ($k$-dimensional) variable from $K$.

By (2.1) and the properties of $f$,

$$f_i(x^{(n)}) \leqq f_i(w) \quad \text{and} \quad f_i(z) \leqq f_i(y^{(n)}) \leqq f_i(w)$$

for all $i$ and $n$. Since $z \neq w$, $f(z) \neq f(w)$, so there is a nonempty set $A$ of indices in $\{1, \cdots, m\}$ such that

$$f_i(z) < f_i(w) \quad \forall i \in A,$$

$$f_i(z) = f_i(w) \quad \forall i \in \{1, \cdots, m\} \setminus A.$$

By this and (2.1)

$$f_i(x^{(n)}) \leqq f_i(w) = f_i(y^{(n)}) = f_i(z), \qquad i \in \{1, \cdots, m\} \setminus A \tag{2.4}$$

holds for every $n$. Let $2\varepsilon = \min\{(f_i(w) - f_i(z)) : i \in A\} > 0$. By (2.3) and continuity of each $f_i$ at $z$ and at $w$, if $n$ is large enough we have

$$|f_i(x^{(n)}) - f_i(z)| < \varepsilon \quad \text{and} \quad |f_i(y^{(n)}) - f_i(w)| < \varepsilon \quad \forall i,$$

so if $i \in A$ then

$$f_i(x^{(n)}) - f_i(y^{(n)}) < (f_i(z) + \varepsilon) - (f_i(w) - \varepsilon) = 2\varepsilon - (f_i(w) - f_i(z)) \leqq 0,$$

so $f_i(x^{(n)}) < f_i(y^{(n)})$; with (2.4) this shows that $f(x^{(n)}) \leqq f(y^{(n)})$, that is, $x^{(n)} \sqsubseteq y^{(n)}$ for large $n$, contradicting (2.2). □

**3. Applications to geometric figures.** In this section we show how to apply Theorem 1 to obtain results about the classes of rectangles, right circular cylinders, and (isosceles) triangles. We also show how to "lift" our results from low to high dimensions.

Let $\mathscr{F} = \{F_i : i \in I\}$ be a family of nonempty geometric objects in $\mathscr{R}^K$, and let $[F_i] = \{Y \in \mathscr{R}^K : Y \text{ is isometric to } F_i\}$. As before, $F_i \sqsubseteq F_j$, or $[F_i] \sqsubseteq [F_j]$, means $F_i$ is containable in $F_j$. We consider three cases of families of geometric objects:

Rectangles. Define $R(W, L) = \{(x, y) \in \mathscr{R}^2 : |x| \leqq W/2, |y| \leqq L/2\}$, and let $\mathscr{F}(R) = \{R(W, L) : 0 < W \leqq L\}$. Let $\sqsubseteq(R)$ denote the containability partial order for $\mathscr{F}(R)$, as well as the induced partial order on the cone

$$K = \{(W, L) \in \mathscr{R}^2 : 0 < W \leqq L\}$$

defined by $(W_1, L_1) \sqsubseteq (R)(W_2, L_2)$ if and only if $R(W_1, L_1) \sqsubseteq (R)R(W_2, L_2)$. The meaning of $\sqsubseteq(R)$ will be clear from its context.

Right circular cylinders. Define $C(W, L) = \{(x, y, z) \in \mathscr{R}^3 : x^2 + y^2 \leqq (W/2)^2, |z| \leqq L/2$ and let $\mathscr{F}(C) = \{C(W, L) : 0 < W \leqq L\}$; note that $\mathscr{F}(C)$ represents only a subset of the set of right circular cylinders. Let $\sqsubseteq(C)$ denote the containability partial order on $\mathscr{F}(C)$, as well as the induced partial order on $K$, with $(W_1, L_1) \sqsubseteq (C)(W_2, L_2)$ if and only if $C(W_1, L_1) \sqsubseteq (C)C(W_2, L_2)$. The meaning of $\sqsubseteq(C)$ will be clear from its context.

Isosceles triangles. Define

$$T(W, H) = \{(x, y) \in \mathscr{R}^2 : 0 \leqq y \leqq H, |x| \leqq (W/2)(1 - y/H)\}$$

and let $\mathscr{F}(T) = \{T(W, H) : 0 < W, 0 < H\}$. Let $\sqsubseteq(T)$ denote the containability partial order on $\mathscr{F}(T)$, as well as the induced partial order on the cone $\mathscr{R}^2_+ = \{(W, H) \in \mathscr{R}^2 : 0 < W, 0 < H\}$ defined by $(W_1, L_1) \sqsubseteq (T)(W_2, L_2)$ if and only if $T(W_1, L_1) \sqsubseteq (T)T(W_2, L_2)$. The meaning of $\sqsubseteq(T)$ will be clear from its context.

We shall now outline the proof of the rectangle containability theorem in [12] in such a way that Theorem 1 is the central device; this will provide a model for some of the later proofs. Clearly, $(K, \sqsubseteq(R))$ satisfies (M) and (H). It only remains to provide points in $K$ that satisfy (2.1)–(2.3). This is accomplished by analyzing the "containability" curve of the square with $W = L = 1$, this curve being defined as the graph of $f(W) = \max\{H | (W, H) \sqsubseteq (R)(1, 1)\}$. The shaded portion of Fig. 1 along with its boundary can be shown to correspond exactly to those rectangles that are containable in the unit

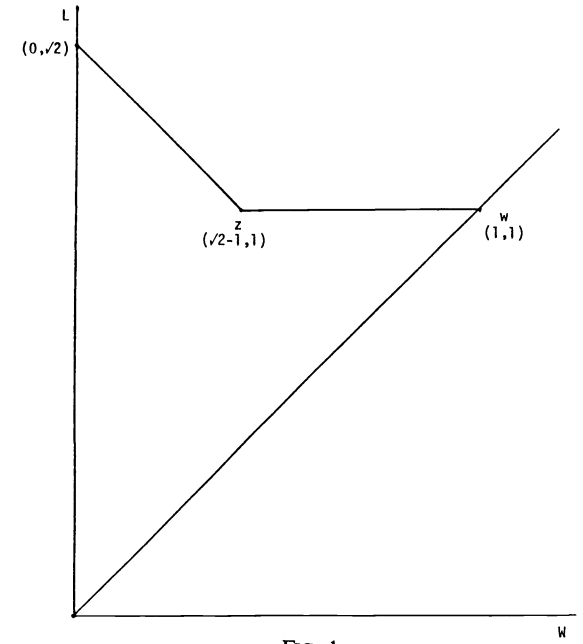


FIG. 1

square. Let $z = (\sqrt{2} - 1, 1)$ and $w = (1, 1)$; the containability curve consists of two line segments, the first joining $(0, \sqrt{2})$ to $z$ and the second joining $z$ to $w$. It is shown in [12] that (2.1) to (2.3) are satisfied for this $z$ and $w$ and for $\{x^{(n)}\} \twoheadrightarrow z$ a sequence of points on the left-hand line segment of the containability curve and $\{y^{(n)}\} \twoheadrightarrow w$ a sequence of points on the right-hand line segment of the containability curve. Thus, the dimension of the rectangular relation is nonfinite.

To obtain the desired result for cylinders, we first prove Theorem 2.

THEOREM 2. *The partial orders* $(K, \sqsubseteq(R))$ *and* $(k, \sqsubseteq(C))$ *are identical.*

*Proof.* Let $(W_i, L_i) \in K$ be given $(i = 1, 2)$. We shall now demonstrate that $(W_1, L_1) \sqsubseteq (R)(W_2, L_2)$ if and only if $(W_1, L_1) \sqsubseteq (C)(W_2, L_2)$.

First, suppose $(W_1, L_1) \sqsubseteq (C)(W_2, L_2)$. Let $\gamma_1$ and $\gamma_2$ be isometric copies of $C(W_1, L_1)$ and $C(W_2, L_2)$ such that $\gamma_1 \subset \gamma_2$, and let $\pi$ be a plane parallel to the central axes of both $\gamma_1$ and $\gamma_2$. Let $\rho_i$ be the projection of $\gamma_i$ onto $\pi (i = 1, 2)$. Then $\rho_i$ is an isometric copy of $R(W_i, L_i)$ and $\rho_1 \subset \rho_2$, so $(W_1, L_1) \sqsubseteq (R)(W_2, L_2)$.

Conversely, suppose $R(W_1, L_1) \sqsubseteq (R)R(W_2, L_2)$. Without loss of generality we may suppose that $(W_1, L_1)$ is $\leqq$-maximal in $K$ with respect to the rectangular containability property, that is, that the conditions $(W, L) \in K$, $R(W, L) \sqsubseteq (R)R(W_2, L_2)$, and $R(W_1, L_1) \sqsubseteq (R)R(W, L)$ imply $(W, L) = (W_1, L_1)$. From [12], $W_1 \leqq W_2$. If $(W_1, L_1) = (W_2, L_2)$ there is nothing to prove, so we may assume $(W_1, L_1) \neq (W_2, L_2)$. Hence, $\leqq$-maximality of $(W_1, L_1)$ implies that $L_1 > L_2$ and $W_1 < W_2$. $\rho_2 = R(W_2, L_2)$ is drawn on the $xz$ plane as the outer rectangle (see Fig. 2). Reference [12] implies that the contained rectangle $\rho_1$, which is an isometric copy of $R(W_1, L_1)$, will be tilted at some angle $\alpha$ and will have its vertices on the interiors of the four sides of $R(W_2, L_2)$, as illustrated. Noting that

(3.1) $$L_1 \sin \alpha + W_1 \cos \alpha = L_2 \geqq W_2 = L_1 \cos \alpha + W_1 \sin \alpha$$

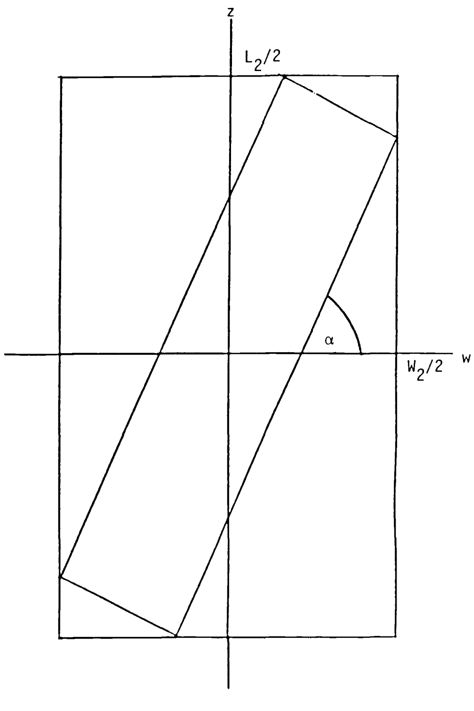


FIG. 2

we obtain $(L_1 - W_1)\sin\alpha \geqq (L_1 - W_1)\cos\alpha$, and hence that

$$\sin\alpha \geqq \cos\alpha, \tag{3.2}$$

an observation that will be used later.

Let $\xi_i$ be the isometric copy of $C(W_i, L_i)$ obtained by rotating $\rho_i$ about its longer central axis. To show that $\xi_1 \sqsubseteq (C)\xi_2$, it suffices to show that each point of $\xi_1$ is a distance no greater than $(W_2/2)$ from the $z$-axis. Since $\xi_1$ is the convex hull of its circular "top" and "bottom" (and here circles do *not* contain their interiors), it suffices to show that each point on these two circles is not more than a distance of $(W_2/2)$ from the $z$-axis. By symmetry we need only treat the top circle. The reader may verify that this top circle consists of precisely the points

(3.3)

$$(x,y,z) = \left(\left(\frac{W_1}{2}\right)\cos\theta\sin\alpha + \left(\frac{L_1}{2}\right)\cos\alpha, -\left(\frac{W_1}{2}\right)\sin\theta, -\left(\frac{W_1}{2}\right)\cos\theta\cos\alpha + \left(\frac{L_1}{2}\right)\sin\alpha\right)$$

for $0 \leqq \theta < 2\pi$. Hence it suffices to show that for each such $(x, y, z)$ we have

$$x^2 + y^2 \leqq \left(\frac{W_2}{2}\right)^2 = \left(\frac{L_1\cos\alpha + W_1\sin\alpha}{2}\right)^2 \tag{3.4}$$

the equality in (3.4) resulting from (3.1). When we multiply and use trigonometric identities and algebra, (3.4) reduces to $W_1\sin^2\theta\cos\alpha \leqq 2L_1\sin\alpha(1 - \cos\alpha)$, and hence to $(1 + \cos\theta)W_1\cos\alpha \leqq 2L_1\sin\alpha$; using (3.2), this inequality is true, so we conclude that $C(W_1, L_1) \sqsubseteq (C)C(W_2, L_2)$, thus concluding the proof. □

An immediate corollary of Theorem 2 is Theorem 3.

THEOREM 3. *The containability partial order for right circular cylinders in $\mathscr{R}^3$ is of nonfinite dimension.*

For the case of triangles, we have Theorem 4.

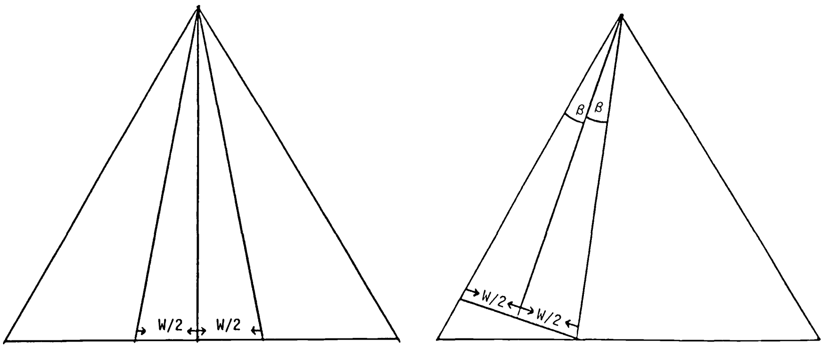
THEOREM 4. *The containability relation for isosceles triangles in $\mathscr{R}^2$ is of nonfinite dimension.*

*Proof.* Clearly, the partial order $(\mathscr{R}^2, \sqsubseteq(T))$ satisfies (M) and (H), so it remains to find points $z$, $w$, $\{x^{(n)}\}$ and $\{y^{(n)}\}$ to satisfy (2.1)–(2.3). The proof will be similar to that for rectangles.

It can be shown by simple geometrical arguments that for a given $W$, $0 < W \leqq 1$, the largest height $h(W)$ for an isosceles triangle containable in the unit equilateral triangle (all sides of length one) is assumed by one of the triangles illustrated in Fig. 3. (Angles will be assumed to be measured in degrees for the remainder of this proof.) In the case of (a) Vertical Orientation, the height is $(\sqrt{3}/2)$, while for (b) Co-lateral Orientation, lengthy but unenlightening computations show that the height is equal to $\sqrt{3}/2)\cos\beta/\cos(30 - 2\beta)$ where $\beta$ is illustrated in Fig. 3(b), and can be shown to satisfy $W = \sqrt{3}\sin\beta/\cos(30 - 2\beta)$. Setting $h(W)$ equal to the maximum for these two candidate heights, we obtain the following:

(3.5) If $W < \sqrt{3}\tan(10)$, then $h(W) = (\sqrt{3}/2)\cos\beta/\cos(30 - 2\beta)$ where $\beta$ satisfies the two relationships $0 < \beta \leqq 10$ and $W = \sqrt{3}\sin\beta/\cos(30 - 2\beta)$.

(3.6) If $W \geqq \sqrt{3}\tan(10)$, then $h(W) = (\sqrt{3}/2)$.



(a) Vertical orientation (b) Co-lateral orientation (Shares side with equilateral triangle)

FIG. 3

The function $h$ is monotonically decreasing over the open interval $(0, \sqrt{3}\tan(10))$, after which it assumes a constant value (see Fig. 4), and the two segments defined by (3.5) and (3.6) meet at the point $z = (\sqrt{3}\tan(10), \sqrt{3}/2)$.

Define the point $w = (1, \sqrt{3}/2)$ that corresponds to the unit equilateral triangle. Let $\{x^{(n)}\} \twoheadrightarrow z$ be a sequence of points in the curved segment of the containability curve defined by (3.5), and let $\{y^{(n)}\} \twoheadrightarrow w$ be a sequence of points on the line segment of the containability curve defined by (3.6). We can verify that $x^{(n)} \sqsubseteq (T)x \sqsubseteq (T)w$ if and only if $x = x^{(n)}$ or $x = w$, and $y^{(n)} \sqsubseteq (T)y \sqsubseteq (T)w$ if and only if $y$ is on the line segment connecting $y^{(n)}$ to $w$. From these observations, it follows that (2.1)–(2.3) hold for $\{x^{(n)}\}$, $\{y^{(n)}\}$, with $z$ and $w$ as defined above. □

We shall close this section with some remarks on lifting negative results from low dimensions to higher dimensions. For example, does the fact that containment of rectangles is not reducible to vector dominance in any $\mathscr{R}^m$ imply a corresponding result for rectangular boxes in $\mathscr{R}^3$? The simplest approach to the box problem seems to use the "local" character of Theorem 1 and the rectangle result: only a small portion of the cone $K = \{(W, L) \in \mathscr{R}^2_+ : W \leqq L\}$ is actually needed, and the widths $W$ in this portion are bounded away from zero.

THEOREM 5. *Let* $\mathscr{F} = \{\mathscr{F}_i : i \in I\}$ *be a nonempty family of geometric objects in* $\mathscr{R}^k$ *for some finite k. Suppose that there is a positive number* $\delta$ *such that, for every line*

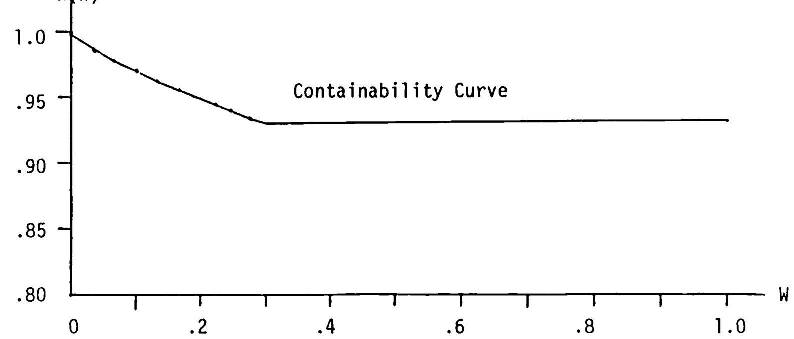


FIG. 4

*L in $\mathscr{R}^k$, every member of $\mathscr{F}$ contains a segment of length $\delta$ parallel to L. Fix d, $0 < d < \delta$, and let $\bar{\mathscr{F}} = \{\bar{F}_i = F_i \times [0, d] \in \mathscr{R}^{k+1} : F_i \in \mathscr{R}^k\}$. Then if $F_1$ and $F_2 \in \mathscr{F}$, we have ($F_1$ is containable in $F_2$) if and only if ($\bar{F}_1$ is containable in $\bar{F}_2$).*

The point here is that a rectangle with sides $d$ and $\delta$ can be placed in one with sides $d$ and $\gamma > \delta$ only with the sides of length $\delta$ lying along those of length $\gamma$. This forces any isometry of $\bar{F}_1$ into $\bar{F}_2$ to be induced in the obvious way by an isometry of $F_1$ into $F_2$. It is clear how the proposition permits one to prove the box result from the rectangle theorem.

**4. An example: quadratic polynomials.** Not all interesting consequences of Theorem 1 involve geometric containment. A natural partial order $\sqsubseteq(P)$ is given on the set of all polynomials with real coefficients by declaring that $P_1 \sqsubseteq (P)P_2$ provided $P_1(x) \leqq P_2(x)$ for all nonnegative real numbers $x$. It is easy to see that the restriction of $\sqsubseteq(P)$ to the class of linear polynomials is reducible to vector dominance in $\mathscr{R}^2$: $A_1X + B_1 \sqsubseteq (P)A_2X + B_2$ if and only if $(A_1, B_1) \leqq (A_2, B_2)$. It is perhaps surprising that this result does not extend one step further to the class $Q$ of quadratic polynomials $P(x) = Ax^2 + 2Bx + C$ with real coefficients $A$, $B$, $C$.

THEOREM 6. *The restriction of the partial order $\sqsubseteq(P)$ to the set Q of quadratic polynomials with real coefficients is of nonfinite dimension.*

*Proof.* We prove the stronger result that $\sqsubseteq(P)$ restricted to the set $Q_+$ of polynomials in $Q$ with strictly positive coefficients $A$, $2B$, $C$ is of nonfinite dimension.

Let $\sqsubseteq(P)$ denote the containability partial order on $Q_+$, as well as the induced partial order on the cone $\mathscr{R}^3_+ = \{(x, y, z) \in \mathscr{R}^3 : x > 0, y > 0, z > 0\}$ defined by

$$(A_1, B_1, C_1) \sqsubseteq (P)(A_2, B_2, C_2)$$

if and only if $A_1X^2 + 2B_1X + C_1 \sqsubseteq (P)A_2X^2 + 2B_2X + C_2$. The meaning of $\sqsubseteq(P)$ will be clear from its context.

Clearly, $(\mathscr{R}^3_+, \sqsubseteq(P))$ satisfies (M) and (H). The reader can easily verify that $(A_1, B_1, C_1) \sqsubseteq (P)(A_2, B_2, C_2)$ precisely if the following two conditions hold:

(4.1) $A_1 \leqq A_2$ and $C_1 \leqq C_2$.

(4.2) Either $B_1 \leqq B_2$ or $(B_1 - B_2)^2 \leqq (A_2 - A_1)(C_2 - C_1)$.

Now let $z = (1, 1, 1)$, $w = (2, 1, 1)$. Let $\varepsilon^{(n)}$ be a sequence of positive numbers such that $\varepsilon^{(n)} < \frac{1}{2}$ and $\varepsilon^{(n)} \to 0$, and let $\delta^{(n)} = (\varepsilon^{(n)}(1 - \varepsilon^{(n)}))^{1/2}$. Set $x^{(n)} = (1 + \varepsilon^{(n)}, 1 + \delta^{(n)}, 1 - \varepsilon^{(n)})$, and $y^{(n)} = (2 - \varepsilon^{(n)}, 1, 1)$. A short computation shows that (2.1)–(2.3) hold for $z$, $w$, $x^{(n)}$, and $y^{(n)}$ as defined above. □

This theorem can be interpreted as a result about the inclusion relationship for the family of plane sets $E_{A,B,C} = \{(x, y) \in \mathscr{R}^2 : x \geqq 0, y \leqq Ax^2 + 2Bx + C\}$; it says that this inclusion partial order is of nonfinite dimension.

**5. Containment for compact sets.** In § 4 of [12] it is shown that the family of congruence classes of plane rectangles can be mapped into $\mathscr{R}^N$, the space of infinite sequences of real numbers, in a manner that converts containability to vector dominance, and that this can be accomplished continuously if convergence in $\mathscr{R}^N$ is taken coordinatewise. We now extend this result substantially.

Let $(X, d)$ be any metric space and let $\mathscr{F}$ denote the family of all nonempty compact subsets of $X$. If $E_1$ and $E_2$ are nonempty bounded subsets of $X$ define

$$\rho_1(E_1, E_2) = \sup_{x \in E_1} [\inf_{y \in E_2} d(x, y)],$$

$$\rho(E_1, E_2) = \max\{\rho_1(E_1, E_2), \rho_1(E_2, E_1)\}.$$

Thus $\rho_1(E_1, E_2) = 0$ if $E_1$ is contained in the closure of $E_2$, and $(\mathcal{F}, \rho)$ is a metric space. $\rho$ is a good measure of closeness of compact sets.

Suppose the topology of $(X, d)$ has a countable base (equivalently, $(X, d)$ is separable), as is the case for $\mathcal{R}^k$ ($k$ a positive integer). Let $(L_n)_{n \geqq 1}$ be an enumeration of all finite unions of members of a fixed countable base for the topology of $X$ which consists of bounded open sets. Then

$$f = (f_1, f_2, \cdots) : \mathcal{F} \to \mathcal{R}^N$$

given by

$$f_n(F) = \rho_1(F, L_n)$$

converts containment—*not* containability—to vector dominance in a continuous manner.

To deal with containability, let $\sim$ denote the relation "is isometric to" on the family of all nonempty bounded subsets of $X$ and let $[E] = \{E' : E' \sim E\}$ for a nonempty bounded subset $E$ of $X$. Define

$$\tilde{\rho}([E_1, E_2]) = \inf\{\rho(E_1', E_2') : E_i' \sim E_i\}$$

for $E_1$, $E_2$ bounded and

$$\tilde{f}_n([F]) = \inf\{\rho_1(F', L_n') : F' \sim F, L_n' \sim L_n\}$$

for $F$ compact. $\tilde{\rho}$ is not in general a metric on the set of isometry classes of compact subsets of $X$, although it is if $(X, d)$ is $\mathcal{R}^k$ with the usual metric. Nonetheless, $\tilde{\rho}$ is a reasonable measure of the "distance" between two isometry classes, and $\tilde{f} = (\tilde{f}_1, \tilde{f}_2, \cdots)$ converts containability to vector dominance; that $\tilde{f}(F_1) \leqq \tilde{f}(F_2)$ implies $F_1 \sqsubseteq F_2$ is a somewhat subtle exercise. In general $\tilde{f}$ is not "continuous," but again if $(X, d)$ is $\mathcal{R}^k$ then

$$|\tilde{f}_n(F_1) - \tilde{f}_n(F_2)| \leqq \tilde{\rho}([F_1], [F_2]),$$

so in this case $\tilde{f}$ is continuous. The central feature of $\mathcal{R}^k$ involved here is that any isometry of one subset of $\mathcal{R}^k$ onto another extends to an isometry of $\mathcal{R}^k$ onto itself.

**Acknowledgment.** Stuart Sidney is grateful to the Faculty of Administration and the Department of Computer Science at the University of Ottawa, Canada, and the School of Computer Science at Carleton University, Ottawa, whose hospitality expedited this collaboration; and to the Department of Mathematics of the University of Hawaii at Manoa, for providing a suitable environment for the completion of this work.

## REFERENCES

[1] N. ALON AND E. SCHEINERMAN, *Degrees of freedom versus dimension for containment orders*, Order, 5 (1988), pp. 11–16.

[2] B. S. BAKER, S. J. FORTUNE, AND S. R. MAHANEY, *Polygon containment under translation*, J. Algorithms, 7 (1986), pp. 532–548.

[3] B. M. CHAZELLE, *The polygon containment problem*, Adv. Comput. Res., 1 (1983), pp. 1–33.

[4] H. EDELSBRUNNER AND M. H. OVERMARS, *On the equivalence of some rectangle problems*, Inform. Process. Lett., 14 (1982), pp. 124–127.

[5] P. C. FISHBURN, *Interval Orders and Interval Graphs, a Study of Partially Ordered Sets*, John Wiley, New York, 1985.

[6] P. C. FISHBURN AND W. T. TROTTER, *Angle orders*, Order, 1 (1985), pp. 333–343.

[7] M. C. GOLUMBIC, *Containment and intersection graphs*, IBM Scientific Center, Israel, T.R. 135, 1984.

[8] H. T. KUNG, F. LUCCIO, AND F. P. PREPARATA, *On finding the maxima of a set of vectors*, J. Assoc. Comput. Mach., 22 (1975), pp. 469–476.

[9] D. G. Kirkpatrick and R. Seidel, *Output-size sensitive algorithms for finding maximal vectors*, Proc. 1st Symposium on Computing Geometry, June 1985, pp. 89–96.

[10] D. T. Lee and F. P. Preparata, *An improved algorithm for the rectangle enclosure problem*, J. Algorithms, 3 (1982), pp. 218–224.

[11] J.-R. Sack, N. Santoro, and J. Urrutia, *Containment of elementary geometric objects*, Proc. 15th SE Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, 1984, pp. 139–146.

[12] N. Santoro, J. B. Sidney, S. J. Sidney, and J. Urrutia, *Geometric containment and vector dominance*, Theoret. Comput. Sci., 53 (1987), pp. 345–352.

[13] N. Santoro and J. Urrutia, *Angle orders, regular n-gon orders and the crossing number*, Order, 4 (1987), pp. 209–220.

[14] E. Scheinerman and J. C. Wierman, *On circle containment orders*, 1987.

[15] J. B. Sidney, S. J. Sidney, and J. Urrutia, *Circle orders, n-gon orders and the crossing number of partial orders*, Order, 5 (1988), pp. 1–10.

[16] J. Urrutia, *Partial orders and Euclidian geometry*, in Algorithms and Order, I. Rival ed., Kluwer Academic Publishers, Norwell, MA, 1989, pp. 387–434.

[17] V. Vaishnavi and D. Wood, *Data structures for rectangle containment and enclosure problems*, Computer Graphics and Image Processing, 13 (1980), pp. 372–384.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 255–261, May 1989

# A $k$-TREE GENERALIZATION THAT CHARACTERIZES CONSISTENCY OF DIMENSIONED ENGINEERING DRAWINGS*

PHILIP TODD†

**Abstract.** In the one-dimensional case (vertical or horizontal dimensioning of a parallel-sided object) a consistent engineering drawing is one whose graph is a tree.

To extend this work to two dimensions, we define a new generalization of the $k$-tree, by relaxing the mutual adjacency condition on vertices adjacent to the new vertex in the usual inductive definition of the $k$-tree. We call these graphs $r$-trees. We also define a generalization of the cyclic property of graphs. A graph is $r$-cyclic if it contains a subgraph, all of whose vertices have degree greater than $r$. We prove that $r$-trees are maximal $r$-acyclic graphs.

The graph theory yields an algorithm for detecting consistency in dimensioned drawings.

**Key words.** $k$-trees, constraint propagation, dimensioning

**AMS(MOS) subject classification.** 05C10

**1. Introduction.** Symbolic dimensions imposed on an engineering drawing serve two purposes. First, they should determine a unique geometry. Second, by selecting those of the measurements in a figure that are to be explicitly toleranced, they constrain the order of manufacture of the object depicted. Further, if a part is to be manufacturable using a given tool set, it must be possible to apply a sequence of dimensional and implicit structural constraints of the type appropriate for that tool set. For example, while a triangle dimensioned by the length of its sides is uniquely defined, it would not be possible to manufacture it from these dimensions using a machine tool that is only able to measure distance along a line and angle at a point.

Given a set $T$ of dimension types, we formally define a dimensioned diagram over $T$ as follows.

DEFINITION. A dimensioned diagram over $T$ is the pair $(G, t)$, where $G$ is a multigraph $(V, E)$ and $t$ is a function $E \twoheadrightarrow T$.

The correspondence between this formal definition and an engineering drawing is illustrated in Fig. 1. In this case $T = \{\text{incidence, angle, distance}\}$; $V$ is the set of lines and points labeled 1–6 in the drawing; and $E$ contains one edge $e_1 = \{v_2, v_6\}$ representing the angle, two edges $e_2 = \{v_1, v_3\}$ and $e_3 = \{v_1, v_5\}$ representing the two distances, and six further edges $e_4 \cdots e_9$ representing the incidence relationships "point 1 lies on line 2," "point 1 lies on line 6," etc. $t$ is defined as follows: $t(e_1) = \text{angle}$; $t(e_2) = t(e_3) = \text{distance}$; and $t(e_i) = \text{incidence}$ for $i > 3$.

For a dimensioned diagram where $V$ is a set of vertical lines, and $T = \{\text{parallel distance}\}$, Requicha [6] states that such a dimensioned diagram is consistent if $G$ is a tree. In this paper, we define graph theoretical concepts, which serve an analogous role in characterizing the consistency of more general dimensioned diagrams.

*Notation*. Let $G = (V, E)$ be a graph, and let $U$ be a subset of $V$; denote $\langle U \rangle$ as the subgraph of $G$ induced by $U$, that is $\langle U \rangle$ has vertex set $U$, and edge set consisting of all edges of $G$ with both end vertices in $U$.

If $v$ is a vertex of $G$ then denote $d_G(v)$ as the degree of $v$ in $G$, and denote $E_G(v)$ as the set of edges of $G$ that have $v$ as one endpoint.

* Received by the editors May 5, 1986; accepted for publication (in revised form) September 22, 1988.

† Computer Graphics Department, Computer Research Laboratory, Tektronix, P.O. Box 500 ms 50/321, Beaverton, Oregon 97077.

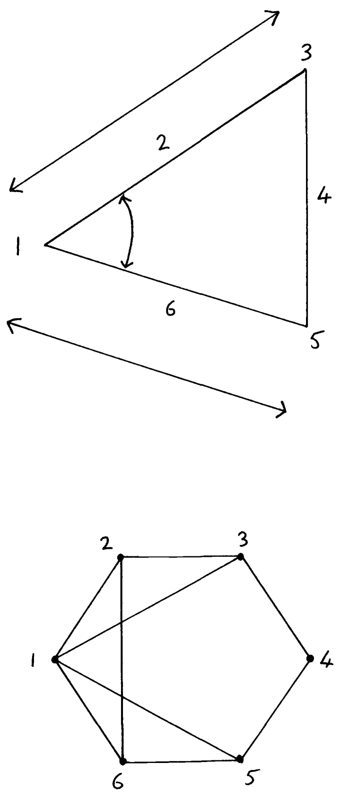


FIG. 1. *The graph corresponding to a triangle dimensioned by two sides and the included angle. Labeled vertices of the graph correspond to labeled points and lines of the triangle. Edges of the graph correspond to the six implicit point-lies-on-line relationships and three explicit dimensions.*

DEFINITION. Given a symmetric binary relation $C$ over $T$ we define the dimensioned diagram $D$ to be $(2, C)$-constructible if and only if there exists an ordering $v_1, v_2, \cdots, v_n$ of $V$ such that we have the following:

(i) $\{v_1, v_2\} \in E$.

(ii) For $2 \leqq i \leqq n$, $d_{\langle\{v_1,\cdots,v_i\}\rangle}(v_i) = 2$, and if $E_{\langle\{v_1,\cdots,v_i\}\rangle}(v_i) = \{e_1, e_2\}$, then $(t(e_1), t(e_2)) \in C$.

If $V$ comprises points and lines in the plane (each of which has two degrees of freedom), and each edge of $E$ represents a constraint comprising of a single equation, then the formal notion of $(2, C)$-constructibility can correspond to geometric constructibility using elementary construction steps combining pairs of constraints from $C$. Suitable choice of $C$ will allow our formal definition of $(2, C)$-constructibility to correspond to the engineering property that the part described by the dimensioned diagram is unambiguously manufacturable from the diagram using a given tool set.

For example, if $T = \{$incidence, angle, distance between points$\}$, $C$ might contain the following pairs: (incidence, incidence), (incidence, angle), (angle, incidence), (incidence, distance), (distance, incidence). This would reflect a manufacturing setup where it is possible to construct the following:

(i) A line through two points or a point through two lines—this corresponds to the (incidence, incidence) pair.

(ii) A line at a given angle to another line through a given point—this corresponds to the (incidence, angle) pair.

(iii) A point on a line a certain distance from another point—this corresponds to the (incidence, distance) pair.

A different manufacturing setup might include (distance, distance) in addition, but as two angles do not specify a line, (angle, angle) would never be an element of $C$.

In § 2 we develop some graph theoretical results that yield necessary conditions for a dimensioned diagram $D = (G, t)$, to be $(2, C)$-consistent for any $C$, in the case where $G$ is a graph (rather than a multigraph).

**2. *r*-Trees.** A higher-dimensional analogue of a graph theoretical tree, called a $k$-tree, is inductively defined [3], [5] as follows. The complete graph on $k$ vertices is a $k$-tree, and a $k$-tree with $n + 1$ vertices is obtained from a $k$-tree with $n$ vertices by adding a vertex adjacent to $k$ mutually adjacent old vertices.

We define an $r$-tree similarly, but relaxing the condition that the old vertices adjacent to our new vertex be mutually adjacent.

DEFINITION. The complete graph on $r$ vertices is an $r$-tree, and an $r$-tree with $n + 1$ vertices is obtained from an $r$-tree with $n$ vertices by adjoining a new vertex adjacent to $r$ old vertices.

Figure 2 shows the smallest $r$-tree that is not a $k$-tree.

A necessary condition that a dimensioned diagram whose representation is a graph (rather than a multigraph) be manufacturable is that its graphical representation is a 2-tree (in the $r$-tree sense).

We define a graph to be $r$-cyclic if it contains a subgraph whose vertices are all of degree greater than $r$. We define a graph to be $r$-acyclic if it is not $r$-cyclic. Clearly, 1-cyclic is equivalent to cyclic. We now prove that an $r$-tree is a maximal $r$-acyclic graph. We use the following results.

LEMMA 1. *A graph with $n \geqq r$ vertices and more than $rn - r(r + 1)/2$ edges is $r$-cyclic.*

*Proof.* The proof is by induction on the number of vertices. The result is clearly true for graphs with at most $r + 2$ vertices, as the only such graph with enough edges is the complete graph on $r + 2$ vertices, which is $r$-cyclic. Now assume the result for graphs with $n$ vertices and let $G$ be a graph with $n + 1$ vertices. If all vertices of $G$ have degree greater than $r$, then $G$ is $r$-cyclic; otherwise look at the subgraph generated by removing a vertex of degree at most $r$. This subgraph has one less vertex and no more than $r$ fewer edges, so is $r$-cyclic by the induction hypothesis. The result follows.

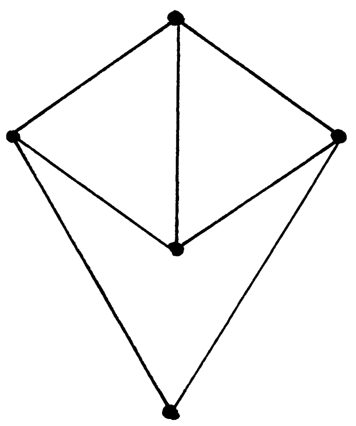


FIG. 2. *This graph is an r-tree but not a k-tree (where $k = r = 2$).*

LEMMA 2. *If $G$ is $r$-acyclic with at least $r + 1$ vertices and has $rn - r(r + 1)/2$ edges, then all vertices of $G$ have degree at least $r$.*

*Proof.* If $G$ has $r + 1$ vertices and the required number of edges, then it is isomorphic to the complete graph on $r + 1$ vertices. This has no vertices degree less than $r$. For graphs with more than $r + 1$ vertices, the subgraph generated by removing a vertex of degree less than $r$ is $r$-cyclic by Lemma 1; hence there can be no such vertex.

THEOREM 3. *If $G$ has at least $r$ vertices, the following are equivalent:*

(i) *$G$ is an $r$-tree.*

(ii) *$G$ is $r$-acyclic with $rn - r(r + 1)/2$ edges.*

*Proof.* (i) $\Rightarrow$ (ii). Clearly, if $G$ is an $r$-tree then $G$ has $rn - r(r + 1)/2$ edges. Now if there is some subgraph of $G$, all of whose vertices have degree greater than $r$, look at the last vertex of the subgraph added in the inductive construction of $G$. This must have degree no greater than $r$ in the subgraph, which is a contradiction from which the result follows.

(ii) $\Rightarrow$ (i). We prove by induction on the number of vertices. The result is true for a graph with $r$ vertices, as this graph is complete. We assume the result for a graph with $n$ vertices and let $G$ be a graph with $n + 1$ vertices satisfying (ii). By definition and Lemma 2 there is a vertex of degree $r$. The graph generated by removing this point is an $r$-tree by the inductive hypothesis, and hence so is $G$.

COROLLARY 4. *If we remove a vertex of degree $r$ from an $r$-tree, the subgraph generated by the new vertex set is also an $r$-tree.*

We thus have the following simple algorithm for deciding whether a graph is an $r$-tree: First ensure there are the required number of edges, then successively remove vertices of degree $r$ until a subgraph with no such vertices is encountered. If this subgraph is the complete graph on $r$ vertices, the original graph is a tree; otherwise it is not. The algorithm is depicted in Fig. 3.

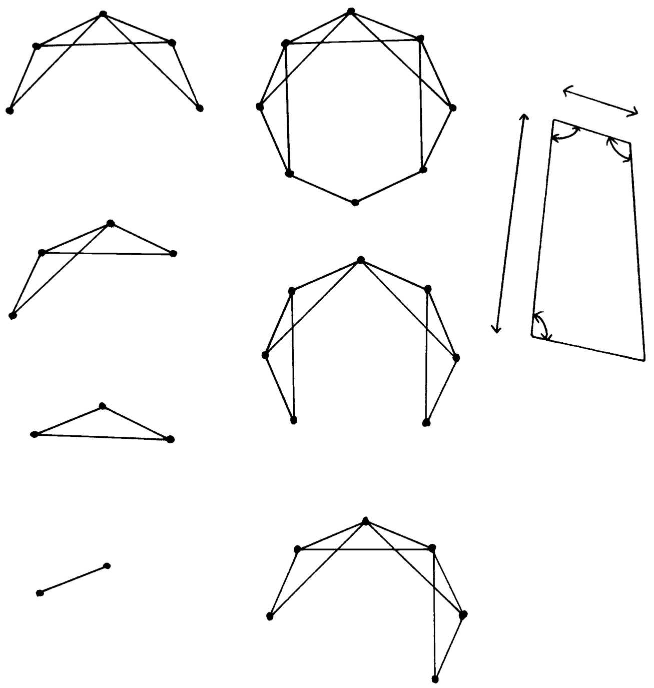


FIG. 3. *A consistently dimensioned quadrilateral and the algorithm applied to its graph.*

THEOREM 5. *If we remove a vertex of degree less than or equal to r from an r-cyclic graph, then the subgraph generated by the new vertex set is also an r-cyclic graph.*

*Proof.* If $G$ is $r$-cyclic, there exists a subgraph $G'$ of $G$ such that all vertices of $G'$ have degree greater than $r$ in $G'$. Any vertex $v$ of $G$ with degree $r$ or less is not a member of $G'$, and hence $G'$ is a subgraph of $H = \langle V \setminus \{v\} \rangle$; hence $H$ is also $r$-cyclic.

Thus we have a simple algorithm for deciding whether a graph is $r$-cyclic: successively remove vertices of degree less than or equal to $r$ until no such vertices remain. If the remaining vertex set is nonempty, the original graph is $r$-cyclic; otherwise it is not.

As sub $r$-trees are simply $r$-acyclic graphs, the above algorithm gives us a method of determining whether a graph can be built into an $r$-tree by adding edges.

**3. (2, $C$)-constructibility.** In this section we develop an analogous theory to the graph theory of the previous section for $(2, C)$-constructible dimensioned diagrams, still with the assumption that the underlying multigraph of the dimensioned diagram should be strictly a graph.

DEFINITION. Let $D = (G, t)$ be a dimensioned diagram over $T$, and let $C$ be a symmetric binary relation on $T$; then an element $v$ of the vertex set of $G$ is $(2, C)$-eliminable in $G$ if $d_G(v) < 2$ or $d_G(v) = 2$ and if $E_G(v) = \{e_1, e_2\}$, then $(t(e_1), t(e_2)) \in C$.

DEFINITION. A dimensioned diagram $D = (G, t)$ is $(2, C)$-cyclic if $G$ contains a subgraph $H$, none of whose vertices are $(2, C)$-eliminable.

DEFINITION. A dimensioned diagram that is not $(2, C)$-cyclic is $(2, C)$-acyclic.

Lemmas 6 and 7 follow immediately from the definitions.

LEMMA 6. *If $D = (G, t)$ is a dimensioned diagram and $G$ is 2-cyclic, then $D$ is $(2, C)$-cyclic.*

LEMMA 7. *If $D = (G, t)$ is $(2, C)$-constructible, then $G$ is a 2-tree.*

LEMMA 8. *If $v$ is $(2, C)$-eliminable in $G$ and $H$ is a subgraph of $G$, then $v$ is $(2, C)$-eliminable in $H$.*

*Proof.* $v$ has degree two or less in $G$ and therefore has degree two or less in $H$. If $v$ has degree two in $H$, then $E_H(v) = E_G(v)$; thus $v$ is $(2, C)$-eliminable.

THEOREM 9. *If $D = (G, t)$ is a dimensioned diagram and $G$ has at least $r$ vertices, then the following are equivalent:*

(i) *$D$ is $(2, C)$-constructible.*

(ii) *$D$ is $(2, C)$-acyclic with $2n - 3$ edges.*

*Proof.* (i) $\Rightarrow$ (ii). If $D$ is $(2, C)$-constructible, then $G$ is a 2-tree by Lemma 7; thus $G$ has $2n - 3$ edges. As $D$ is $(2, C)$-constructible, there is an ordering $v_1, v_2, \cdots, v_n$ of the vertices of $V$ such that for $2 < i \leqq n$, $v_i$ is eliminable in $\langle \{v_1, v_2, \cdots, v_i\} \rangle$. Now assume there is some subgraph $H$ of $G$, none of whose vertices is $(2, C)$-eliminable, and let $v_k$ be the last vertex of the subgraph to appear in the above ordering. As $H \subset \langle \{v_1, v_2, \cdots, v_k\} \rangle$, $v_k$ must be $(2, C)$-eliminable in $H$ by Lemma 8, a contradiction from which the result follows.

(ii) $\Rightarrow$ (i). We prove (ii) $\Rightarrow$ (i) by induction on the number of vertices. The result is trivially true for a dimensioned diagram with two vertices. We assume the result for a diagram with $n$ vertices and let $D$ be a diagram with $n + 1$ vertices satisfying (ii). By definition, Lemma 6, and Lemma 2, $D$ contains a $(2, C)$-eliminable vertex of degree two. The diagram generated by removing this point is $(2, C)$-constructible by the inductive hypothesis, and hence so is $D$.

COROLLARY 10. *If we remove a $(2, C)$-eliminable vertex from a $(2, C)$-constructible diagram, then the diagram generated by the new vertex set is also $(2, C)$-eliminable.*

THEOREM 11. *If we remove a $(2, C)$-eliminable vertex from a $(2, C)$-cyclic diagram, then the diagram generated by the new vertex set is also $(2, C)$-cyclic.*

*Proof.* If $D$ is $(2, C)$-cyclic, then there exists a subgraph $G'$ of $G$ such that no vertex of $G'$ is $(2, C)$-eliminable in $G'$. Any $(2, C)$-eliminable vertex $v$ of $G$ is not a member of $G'$, and hence $G'$ is a subgraph of $H = \langle V \setminus \{v\} \rangle$; hence the diagram generated by $H$ is also $(2, C)$-cyclic.

As a result of the above theorems, we have algorithms for detecting $(2, C)$-constructible and $(2, C)$-cyclic dimensioned diagrams. To detect $(2, C)$-cyclic diagrams, remove eliminable vertices until there are no more. If we are left with a nonempty vertex set, the original graph is $(2, C)$-cyclic; otherwise it is $(2, C)$-acyclic. To detect $(2, C)$-constructible diagrams, check there are $2n - 3$ edges, and then make sure the graph is $(2, C)$-acyclic as above.

**4. Multigraph 2-trees.** In §§ 2 and 3 we restricted our discussion to dimensioned diagrams whose underlying multigraphs are strictly graphs. In this section we identify those multigraph dimensioned drawings for which the algorithms developed above still work.

Multigraphs arise naturally in dimensioned diagrams in representing the parallel distance dimension type. Parallel distance represents a pair of constraints between two lines, and is most appropriately represented as a pair of edges between a single pair of vertices of the underlying graph.

Figure 4 shows that our 2-tree detection algorithm does not extend intact to multigraphs. If point $B$ is removed first the algorithm runs through to completion; if point $A$ is removed first, the algorithm sticks.

In this section, we constrain the definition of a multigraph 2-tree in such a way that our algorithm still works. Fortunately, the constraint is a natural one from the dimensioned diagram viewpoint.

We define a multigraph 2-tree inductively as follows. A graph comprising two vertices and one edge is a multigraph 2-tree. A 2-tree with $n + 1$ vertices may be obtained from a 2-tree with $n$ vertices by adding one vertex and two edges adjacent to that vertex, with the constraint that if two old vertices are adjacent to the new vertex then there is no chain of double edges joining the two old vertices.

THEOREM 12. *A multigraph with more than two vertices is a 2-tree if and only if it has a legal vertex of degree two and the graph generated by removing any vertex of degree two is a 2-tree.*

*Proof.* Sufficiency is obvious. To prove necessity we do the following. If $G$ is a multigraph 2-tree, there exists a vertex ordering $v_1, v_2, \cdots, v_n$ where each $v_i$ for $i > 2$ is the new vertex added to subgraph $v_1, \cdots, v_{i-1}$ in a way consistent with the inductive definition.

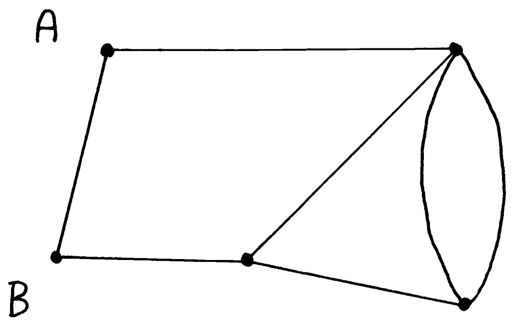


FIG. 4. *This example graph shows some care is necessary in applying our algorithm to multigraphs. If vertex B is removed first, the algorithm runs through to completion; if vertex A is removed first it sticks.*

Clearly $v_n$ is a legal vertex of degree two. Now let $v_i$ be a general vertex of degree two, and consider the subgraph generated by removing it. There are three possible cases, in each case we display an ordering of the vertices of the subgraph that satisfies the inductive definition:

(i) If $i = n$, then the subgraph is a 2-tree by construction.

(ii) If $2 < i < n$, then the sequence $v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n$ gives a legitimate 2-tree construction for the subgraph.

(iii) $i = 1$ or 2 (without loss of generality assume $i = 1$): If $G$ has only three vertices, then the result is trivial. Otherwise $v_1$ is adjacent to $v_2$ and one $v_j$ for $2 < j \leqq n$.

(a) If $j = 3$, then $v_2$ and $v_3$ are joined by a single edge and $v_2, v_3, \cdots, v_n$ gives a legitimate 2-tree construction for the subgraph.

(b) If $j > 3$, then a simple inductive argument using the requirement that a new vertex may not be joined to two vertices in a chain of double edges shows that $v_2, \cdots, v_j$ must be connected by a chain of double edges. Hence the order $v_j, v_{j-1}, \cdots, v_2, v_{j+1}, \cdots, v_n$ gives a legitimate 2-tree construction for the subgraph.

In the systems we have discussed, double edges only appear representing parallel distance dimensions, from a dimensioning point of view the restriction in the definition of a multigraph 2-tree prevents construction of a line or point from dimensions relating its position to that of two parallel lines. This restriction is geometrically natural.

**5. Conclusion.** We have defined a new class of treelike graphs by relaxing the mutual adjacency condition on vertices adjacent to the new vertex in the usual inductive definition of the $k$-tree. We call these graphs $r$-trees. We have also defined a generalization of the cyclic property of graphs. A graph is $r$-cyclic if it contains a subgraph, all of whose vertices have degree greater than $r$.

The problem of identifying partial $k$-trees is hard [1], [2]. By contrast partial $r$-trees are simply $r$-acyclic graphs and may be identified by the algorithm given at the end of § 2.

For $r = 2$, $r$-trees correspond to consistently dimensioned engineering drawings. The maximal $r$-acyclic graph property yields an algorithm for detecting consistency and manufacturability of a dimensioned drawing. The algorithm also gives an ordering of the graph vertices that may be interpreted (in reverse) as a geometric construction sequence for the drawing.

## REFERENCES

[1] S. Arnborg and A. Proskurowski, *Characterization and recognition of partial k-trees*, Congr. Numer., 47 (1985), pp. 69–75.

[2] ———, *Characterization and recognition of partial* 3-*trees*, SIAM J. Algebraic Discrete Meth., 7 (1986), pp. 305–314.

[3] L. W. Beineke and R. E. Pippert, *On the number of k-dimensional trees*, J. Combin. Theory, 6 (1969), pp. 200–205.

[4] A. K. Dewdney, *Higher-dimensional tree structures*, J. Combin. Theory Ser. B, 17 (1974), pp. 160–169.

[5] F. Harary and E. M. Palmer, *On acyclical simplicial complexes*, Mathematika, 15 (1968), pp. 112–115.

[6] A. Requicha, *Part and assembly description languages*: I—*dimensioning and tolerancing*, Technical Report TM-19, Technical Automation Project, University of Rochester, New York, 1977.

SIAM J. DISC. MATH.
Vol. 2, No. 2, pp. 262–283, May 1989

# THE RADON TRANSFORM ON $\mathbb{Z}_n$*

JAMES ALLEN FILL†

**Abstract.** The Radon transform on $\mathbb{Z}_n$ that arises in the analysis of directional data and circular time series replaces each value $f(k)$ of a function $f$ by the average value of $f$ over the translate of a set $S$ by $k$. For general $S$ the discrete Fourier transform is used to characterize the null space and range of the transform and to calculate a (generalized) inverse transform. Explicit forms of the coefficients in the inversion formula are obtained in the two cases $S = \{-r, +r\}$ and the symmetric moving average $S = \{-r, \ldots, +r\}$. We show that the proportion of all choices $S$ of size $t$ giving invertible transforms is nearly unity when $\min(t, n-t)$ is large.

**Key words.** Radon transform, Fourier analysis, Moore-Penrose generalized inverse, circulant, circular time series, moving average

**AMS(MOS) subject classifications.** 44A15, 15A09, 42A16, 15A03

## 1. Introduction.

**Notation.** Throughout this paper, in order to avoid notational conflict with (1.1) below, we shall denote the complex conjugate of a number $z$ by $z^*$. More generally, $M^*$ is the conjugate transpose of the matrix $M$. The size of a set $S$ is denoted $\#S$. The characteristic function, i.e., indicator function, of $S$ is written $\chi_S$. The greatest common divisor of integers $j_1, \ldots, j_k$ is written $(j_1, \ldots, j_k)$. For real $x$, the largest integer no larger than $x$ (respectively, the smallest integer no smaller than $x$) is written $\lfloor x \rfloor$ (respectively, $\lceil x \rceil$).

The study of directional data and circular time series is concerned with functions $f : \mathbb{Z}_n \to \mathbb{R}$ or $f : \mathcal{U} \to \mathbb{R}$, where $\mathbb{Z}_n$ is the group of integers with addition modulo $n$ and $\mathcal{U} = [0, 1)$ is the (continuous) circle with addition modulo one. On $\mathbb{Z}_n$, we interpret $f(j)$ as the proportion of observations from the circle $\mathcal{U}$ of unit circumference that lie in the range $[j/n, (j+1)/n)$. For example, if the wind direction at a specified location is recorded hourly for 24 hours and—with the circle $\mathcal{U}$ aligned with the compass directions in such a way that $(0/4, 1/4, 2/4, 3/4) = (E, N, W, S)$—$n = 4$, then $f(1)$ is 1/24 of the number of hourly readings lying between due north and due west. (Many interesting examples of circular data can be found in Mardia [10] and Batschelet [2].) In such cases $f$ is a probability mass function (pmf) on $\mathbb{Z}_n$. In the limit as $n \to \infty$ we encounter probability density functions (pdf's) $f$ on $\mathcal{U}$. We shall limit our discussion in this paper to the discrete circle $\mathbb{Z}_n$. Corresponding issues for pdf's on the continuous circle $\mathcal{U}$ are addressed in Fill [6].

In the analysis of (circularly or otherwise) ordered observations we commonly filter the data. On $\mathbb{Z}_n$, this means that $f$ is replaced by $\bar{f} : \mathbb{Z}_n \to \mathbb{R}$ defined by

$$\bar{f}(k) = \sum_{j \in \mathbb{Z}_n} s(k-j) f(j), \qquad k \in \mathbb{Z}_n.$$

When $s$ is the characteristic function $\chi_{-S}$ of some subset $-S$ of $\mathbb{Z}_n$, the corresponding

* Received by the editors June 15, 1987; accepted for publication (in revised form) September 7, 1988.

† Department of Statistics, Stanford University, Stanford, California 94305. Written in part while on leave at The University of Chicago. This manuscript was prepared using computer facilities supported in part by National Science Foundation grants DMS-8601732 and DMS-8404941 to the Department of Statistics at The University of Chicago, Chicago, Illinois 60637.

transformation

$$\overline{f}(k) = \sum_{j \in S+k} f(j) \tag{1.1}$$

is called the *Radon transform* of $f$ based on (translates of) $S$ and is the target of our investigation here. Except for the constant factor $1/\#S$ omitted to simplify the form of our results, $\overline{f}(k)$ is the (unweighted) average value of $f$ over the translate $S+k$ of the set $S$ by $k$, and $\overline{f}$, like $f$, is a pmf.

Two simple examples are presented in Example 2.1 below and later treated in depth: (a) the "shell" $S = \{-r, +r\}$ and (b) the "ball" $S = \{-r, \ldots, +r\}$; the latter case corresponds to the familiar (circular) symmetric moving average filter. Another natural choice is (c) $S = \{kd : k = 0, \ldots, n/d - 1\}$, with $d$ a specified divisor of $n$. This filter might be employed in the following situation. A sample $X_1, \ldots, X_t$ is drawn from a pmf $h$ on $\mathbb{Z}_n$ and the empirical pmf $f$ is computed. Under the hypothesis that $h$ is periodic with period $d|n$, the pmf $\overline{f}/\#S = (d/n)\overline{f}$ is a better estimate of $h$ than is $f$.

Let $S$ be a given subset of $\mathbb{Z}_n$. The fundamental question we address in this paper is this: In what ways are the data in the "processed" form $\overline{f}$ less informative than the original data? Alternatively, given $\overline{f}$, to what extent, and through what calculations, can $f$ be reconstructed from $\overline{f}$?

The answers are obtained through Fourier methods in §2. Since the Radon transform $Tf = \overline{f}$ of (1.1) is linear, the amount of information discarded by the transform can be measured by the nullity (say, $\nu$) of $T$. We show how to relate $\nu$ to the characteristic polynomial $p_S(x) = \sum_{k \in S} x^k$ of $S$ and in §2.1 pursue this connection in detail. We show in particular that when $n$ is prime, $\nu = 0$: for any $S$, the transform (1.1) is invertible. In §2.2 we show how to best choose $f$ in order to approximate a given $g$ by $Tf$ when $\nu \geq 0$ and $g$ is not necessarily in the range of $T$: use $f = Ug$, where $U = T^{\div}$ is the Moore-Penrose generalized inverse to $T$. This might be useful when $\overline{f}$ is recorded with some error as $g$ and reconstruction of $f$ is desired. We use similar calculations to characterize the set $\{g : g = \overline{f}$ for some pmf $f\}$ of interest in the data-based situation described above.

In §3 we use results of Fill [5]–[7] to show that when both $t$ and $n-t$ are large, the Radon transforms based on nearly all subsets of $S$ of size $t$ are invertible. As we show, this result to some extent complements a theorem of Freedman and Lane [8] concerning the empirical distribution of a set of random Fourier coefficients.

When, by contrast, $\#S = t$ is two or three, the nullity $\nu$ can be expressed quite simply in terms of $S$. Fill [7] derives the probability distribution of $\nu$ when $S$ of size $t$ is chosen at random. As a consequence we have the roughly stated conclusion that a proportion 2/3 (respectively, 12/13) of all Radon transforms $T = T_{n,S}$ for which $n \geq 3$ and $\#S$ equals two (respectively, three) are invertible.

In §§4 (invertible $T$) and 5 (noninvertible $T$) we derive explicit formulas for the (respectively, ordinary and generalized) inverse transform $U = T^{\div}$ $(= T^{-1}$ in the invertible case) when $S$ is a shell or a ball. The formulas of these sections are much more useful for considering such questions as the relative size of the coefficients $u(k)$ in $(Ug)(j) = \sum_{k \in \mathbb{Z}_n} u(k) g(j+k)$ than are the corresponding Fourier-derived formulas from §2.

## 2. Fourier analysis for Radon transforms based on translates in $\mathbb{Z}_n$.

We define the *Radon transform* $T \equiv T_S$ based on translates of a given nonempty subset $S$ of $\mathbb{Z}_n$ (the discrete cyclic group with addition modulo $n$) as follows. Given $f : \mathbb{Z}_n \to \mathbb{C}$, define its Radon transform $Tf \equiv \overline{f} : \mathbb{Z}_n \to \mathbb{C}$ by

$$(Tf)(k) = \overline{f}(k) = \sum_{j \in S+k} f(j), \qquad k \in \mathbb{Z}_n. \tag{2.1}$$

In this section we shall use elementary Fourier analysis to determine exactly when the linear transform $T$ is invertible. The analysis will yield the (unique) Moore-Penrose generalized inverse $U = T^{\div}$ (see Davis [3, §§2.8, 2.3]), for background to $T$. When $T$ is invertible, $U$ is simply the (ordinary) inverse transform. Otherwise calculation of $U$ will enable us, for given $g : \mathbb{Z}_n \to \mathbb{C}$, to characterize the set of solutions $f$ to the equation $Tf = g$. When this set is empty, we shall use $U$ to find $f$ minimizing the Euclidean norm $\|Tf - g\|$.

In this paper we shall treat the following two familiar cases in detail.

*Example* 2.1. Let $d(j,k) = \min\{(k-j) \bmod n,\ (j-k) \bmod n\}$ be the usual bi-invariant metric on $\mathbb{Z}_n$. Here, as usual, we write $a \bmod n = b$ if $b \in \{0,\dots,n-1\}$ and $a \equiv b (\bmod\ n)$. Define $d(j) = d(0,j)$. Let $2r+1 \le n$.

(a) Let $S_r := \{j \in \mathbb{Z}_n : d(j) = r\} = \{-r, r\}$ denote the *shell of radius* $r$ centered at zero. Then $T \equiv T_{S_r}$ sums over the two points at distance exactly $r$.

(b) If $S$ is the *ball of radius* $r$ $S_r^+ := \{j \in \mathbb{Z}_n : d(j) \le r\}$ we get the "nearest neighbors" transform that sums $f$ over the $\nu := 2r+1$ points at distance no greater than $r$.

We shall find it convenient to regard each $f : \mathbb{Z}_n \to \mathbb{C}$ as the column vector $f = (f(0), f(1), \dots, f(n-1))'$ in $\mathbb{C}^n$. Then (2.1) can be rewritten $\bar{f} = Tf$, where $T$ is an $n \times n$ matrix with $(k,\ell)$-entry $t_{k\ell} = \chi_S(\ell - k)$, $k \in \mathbb{Z}_n$, $\ell \in \mathbb{Z}_n$. Here $\chi_S$ is the characteristic function of the set $S$, and the subtraction $k - \ell$ is done modulo $n$. In other words $T$ is the circulant matrix

$$T = \operatorname{circ}(\chi_S) \tag{2.2}$$

and the well-developed theory of circulant matrices [3] can be applied to the study of Radon transforms. We shall follow much of the notation of Davis in what follows.

As demonstrated by Lemma 3 in Diaconis and Graham [4], Fourier techniques are useful in discovering how Radon transforms based on translates behave for any finite group. In the present setting the Fourier transform sends $f : \mathbb{Z}_n \to \mathbb{C}$ to $\hat{f} : \mathbb{Z}_n \to \mathbb{C}$ given by

$$\hat{f}(k) = n^{-1/2} \sum_{j=0}^{n-1} f(j) e^{-2\pi i jk/n}$$

with inverse

$$f(k) = n^{-1/2} \sum_{j=0}^{n-1} \hat{f}(j) e^{2\pi i jk/n}.$$

Let $\zeta_n = e^{2\pi i/n}$; then the Fourier transform is represented by the unitary Fourier matrix

$$F = n^{-1/2} (\zeta_n^{-kl})_{k \in \mathbb{Z}_n, \ell \in \mathbb{Z}_n} \tag{2.3}$$

and the inverse transform by $F^*$, the conjugate transpose of $F$.

It is well known that every circulant is diagonalized by $F$. Our fundamental theorem, Theorem 2.2, follows from Theorem 3.2.2 of [3]. Recall that the nullity of a matrix is the dimension of its null space.

THEOREM 2.2. *The matrix $T$ of* (2.2) *has the diagonalization*

$$T = F^* \Lambda F, \qquad \text{i.e., } FT = \Lambda F, \tag{2.4}$$

*where $F$ is the Fourier matrix* (2.3) *and*

$$\Lambda = n^{1/2} \operatorname{diag}(\hat{\chi}_S(-j) : j = 0, \dots, n-1). \tag{2.5}$$

*The eigenvalues of $T$ are therefore not necessarily distinct values*

$$n^{1/2}\hat{\chi}_S(-j) = \sum_{k\in S} e^{2\pi ijk/n}, \qquad j = 0,\ldots,n-1. \tag{2.6}$$

*Regardless of $S$, the columns of $F^*$ are a set of eigenvectors for $T$ and form an orthonormal basis for $\mathbb{C}^n$.*

*In particular $T$ is invertible if and only if*

$$\hat{\chi}_S(j) \neq 0 \quad \textit{for every} \quad j = 0,\ldots,n-1.$$

*More generally, the nullity of $T$ is the number of $j \in \{0,\ldots,n-1\}$ for which $\hat{\chi}_S(j) = 0$.*

Note $\hat{\chi}_S(-j) = \hat{\chi}_S(n-j) = (\hat{\chi}_S(j))^*$.

When $S$ is symmetric about zero, as in Example 2.1 above, then the eigenvalues (2.6) are real and, except for one or two values ($j = 0$ and, if $n$ is even, $j = n/2$) occur in pairs:

$$n^{1/2}\hat{\chi}_S(-j) = n^{1/2}\hat{\chi}_S(j) = \sum_{k\in S} \cos\left(\frac{2\pi jk}{n}\right).$$

The eigenspace corresponding to $n^{1/2}\hat{\chi}_S(j)$ for given $j \neq 0, n/2$ is spanned by the *real* orthonormal vectors $\left((2/n)^{1/2}\cos(2\pi jk/n)\right)_{k\in\mathbb{Z}_n}$ and $\left((2/n)^{1/2}\sin(2\pi jk/n)\right)_{k\in\mathbb{Z}_n}$.

In terms of the Radon transform (2.1), the result (2.4) can be restated

$$\hat{\bar{f}}(j) = n^{1/2}\hat{\chi}_S(-j)\hat{f}(j), \qquad j \in \mathbb{Z}_n. \tag{2.7}$$

The norm of the transform is its largest absolute eigenvalue, namely, $n^{1/2}\hat{\chi}_S(0) = \#S$.

If $\hat{\chi}_S$ never vanishes, then $\hat{f}(j) = n^{-1/2}\hat{\bar{f}}(j)/\hat{\chi}_S(-j)$ for every $j$ (i.e., $T^{-1} = F^*\Lambda^{-1}F$) and we arrive at the following *inversion formula:*

$$\begin{aligned} f(j) &= n^{-1}\sum_{k=0}^{n-1} \frac{\hat{\bar{f}}(k)}{\hat{\chi}_S(-k)}\, e^{2\pi ijk/n} \\ &= \sum_{l=0}^{n-1} \bar{f}(l)\cdot \frac{1}{n}\sum_{k=0}^{n-1} \frac{e^{2\pi ik(j-l)/n}}{n^{1/2}\hat{\chi}_S(-k)}, \qquad j\in\mathbb{Z}_n, \end{aligned} \tag{2.8}$$

for reconstruction of $f$ from $\bar{f}$. The coefficient of each $\bar{f}(l)$ in this expansion is real. The norm of the inverse transform is $1/\min\{|\hat{\chi}_S(j)| : j \in \mathbb{Z}_n\}$.

In studying a noninvertible matrix $T$, the Moore-Penrose generalized inverse (M-P inverse) $T^{\div}$ to $T$ is of great help. This $T^{\div}$ is the unique matrix $U$ satisfying the following: (1) $TUT = T$; (2) $UTU = U$; (3) $(TU)^* = TU$; and (4) $(UT)^* = UT$. As examples, if $T$ is invertible then $T^{\div} = T^{-1}$; if $\Lambda = \operatorname{diag}(\lambda_0,\ldots,\lambda_{n-1})$, then $\Lambda^{\div} = \operatorname{diag}(\lambda_0^{\div},\ldots,\lambda_{n-1}^{\div})$, where $\lambda^{\div}$ is defined as $1/\lambda$ for $\lambda \neq 0$ and as $0$ for $\lambda = 0$.

If $C$ is the circulant $F^*\Lambda F$, then $C^{\div} = F^*\Lambda^{\div}F$, [3, Thm. 3.3.1]. Our matrix $T$ therefore has as M-P inverse the circulant matrix

$$U := T^{\div} = F^*\Lambda^{\div}F \tag{2.9}$$

where $\Lambda$ is given by (2.5)–(2.6). $U$, like $T$, has real entries; if $T$ (i.e., $S$) is symmetric, then so is $U$.

We can rephrase (2.9) by saying that if a transform $U$ is defined in Fourier terms by

$$\widehat{Ug}(j) = \left[n^{1/2}\hat{\chi}_S(-j)\right]^{\div}\hat{g}(j),$$

in other words by

$$(Ug)(j) = n^{-1/2}\sum_{k=0}^{n-1}\left[n^{1/2}\hat{\chi}_S(-k)\right]^{\div}\hat{g}(k)e^{2\pi ijk/n}$$

$$(2.10)\qquad = \sum_{\ell=0}^{n-1} g(\ell)\cdot\frac{1}{n}\sum_{k=0}^{n-1}\left[n^{1/2}\hat{\chi}_S(-k)\right]^{\div} e^{2\pi ik(j-\ell)/n},$$

for $g : \mathbb{Z}_n \to \mathbb{C}$ and $j \in \mathbb{Z}_n$, then $U$ is real, linear, self-adjoint, and rotationally equivariant, commutes with $T$, and satisfies $TUT = T$ and $UTU = U$. Note

$$\widehat{UTf}(j) = \widehat{TUf}(j) = \begin{cases} \hat{f}(j) & \text{if } \hat{\chi}_S(-j) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

By direct observation or from §2.8.5 of Davis [3], $T$ and $U$ have the same range space, and $TU = UT$ is orthogonal projection onto this subspace of $\mathbb{C}^n$. Thus the range $\mathcal{R}(T)$ of $T$ is the null space $\mathcal{N}(I - TU)$ of $I - TU$, and this space in turn equals $\mathcal{R}(TU)$. Therefore

$$(2.11)\qquad g \in \mathcal{R}(T) \quad \text{if and only if} \quad \hat{g}(j) = 0 \text{ whenever } \hat{\chi}_S(-j) = 0.$$

The criterion (2.11) is also directly evident from (2.7). Likewise, $\mathcal{N}(T) = \mathcal{R}(I - UT) = \mathcal{N}(UT) = \mathcal{R}(UT)^{\perp} = \mathcal{R}(TU)^{\perp} = \mathcal{R}(T)^{\perp}$, and

$$(2.12)\qquad f \in \mathcal{N}(T) \quad \text{if and only if} \quad \hat{f}(j) = 0 \text{ whenever } \hat{\chi}_S(-j) \neq 0.$$

Since $U^{\div} = T$, we have symmetrically $\mathcal{R}(U) = \mathcal{R}(T) = \mathcal{N}(T)^{\perp} = \mathcal{N}(U)^{\perp}$. It follows from (2.12) with $j = 0$ that $\sum_{k\in\mathbb{Z}_n} f(k) = 0$ is a necessary condition for $\overline{f} = 0$ whenever $S \neq \emptyset$; this is just as clear from the identity

$$(2.13)\qquad \overline{f}(+) = \#S \cdot f(+)$$

derivable from (2.1) or (2.7), where

$$h(+) := \sum_{j\in\mathbb{Z}_n} h(j)$$

for $h : \mathbb{Z}_n \to \mathbb{C}$.

*Example* 2.3. We return to the two settings of Example 2.1.

(a) *Shells*. The eigenvalues of $T$ are

$$n^{1/2}\hat{\chi}_{S_r}(j) = 2\cos\,(2\pi jr/n).$$

(We have thus reproduced the results of §6.5.2 of Anderson [1].) It follows that the Radon transform based on $S_r$ is invertible if and only if $n$ is *not* a multiple of $2^{\alpha+2}$, where $2^{\alpha}$ is the largest power of two that divides $r$.

Suppose now that $T$ is not invertible. Then the nullity of the transform $T$ is $2(n,r)$: we recognize the $2(n,r)$ vectors

$$f_k(j) = n^{-1/2}\exp^{(-2\pi ij(2k+1)/[4(n,r)])}, \qquad 0 \leq k < 2(n,r)$$

as an orthonormal basis for $\mathcal{N}(T)$. Thus the null space of $T$ consists of all vectors in $\mathbb{C}^n$ periodic of period $4(n,r)$ (which divides $n$) such that the second $2(n,r)$ entries are the respective negatives of the first $2(n,r)$. Since $\mathcal{R}(T) = \mathcal{N}(T)^{\perp}$, it follows that $g \in \mathcal{R}(T)$ if and only if

$$\sum_{\ell=0}^{n/[2(n,r)]-1} (-1)^{\ell} g(j + \ell \cdot 2(n,r)) = 0 \text{ for each } j = 0, \ldots, 2(n,r) - 1. \tag{2.14}$$

Whether or not $T$ is invertible, we obtain the M-P inverse $U$ of $T$ via

$$(Ug)(j) = \sum_{\ell=0}^{n-1} g(\ell) \cdot \frac{1}{2n} \sum_{k=0}^{n-1} \left[\cos\left(\frac{2\pi kr}{n}\right)\right]^{\div} \cos\left(\frac{2\pi k(j-\ell)}{n}\right). \tag{2.15}$$

(b) *Balls.* The eigenvalues of $T$ are $n^{1/2}\hat{\chi}_S(0) = 2r + 1 = \nu$ and

$$n^{1/2}\hat{\chi}_{S_r^+}(j) = \sin\left(\frac{\pi j \nu}{n}\right) \Big/ \sin\left(\frac{\pi j}{n}\right), \qquad j = 1, \ldots, n-1.$$

Thus the Radon transform based on $S_r^+$ is invertible if and only if $n$ and $\nu$ are relatively prime. In any case the nullity of the transform is one less than the greatest common divisor $(n, \nu)$ of $n$ and $\nu$.

We have $\hat{\chi}_S(j) = 0$ precisely when $j = kn/(n,\nu)$ for some $0 < k < (n,\nu)$. According to (2.12), $f \in \mathcal{N}(T)$ if and only if the values of $f$ add to zero (compare (2.13)) and $f$ is periodic of period $(n,\nu)$. Since $\mathcal{R}(T) = \mathcal{N}(T)^{\perp}$, it follows that $g \in \mathcal{R}(T)$ if and only if

$$\sum_{k \equiv j \pmod{(n,\nu)}} g(k) \quad \text{is constant in } j = 0, \ldots, (n,\nu) - 1. \tag{2.16}$$

The M-P inverse $U$ of $T$ is given by

$$(Ug)(j) = \sum_{\ell=0}^{n-1} g(\ell) \cdot \frac{1}{n}\left(\frac{1}{\nu} + \sum_{k=1}^{n-1} \left[\sin\left(\frac{\pi k \nu}{n}\right) \Big/ \sin\left(\frac{\pi k}{n}\right)\right]^{\div} \cos\left(\frac{2\pi k(j-\ell)}{n}\right)\right). \tag{2.17}$$

**2.1. Invertibility: the characteristic polynomial.** In this section we give criteria for the invertibility of the Radon transform based on general $S$ in terms of the prime factorization of $n$. The key to the analysis will be the characteristic polynomial

$$p_S(x) = \sum_{k \in S} x^k$$

of degree at most $n-1$ in $x$. According to Theorem 2.2, the transform is invertible if and only if $p_S$ has no zeros among the $n$ complex $n$th roots of unity, namely,

$$\zeta_n^j = e^{2\pi i j/n}, \qquad j = 0, \ldots, n-1.$$

The standard algebraic number theory results used here can be found in Washington [12].

*Case* 1. $n$ prime. We show that *the Radon transform based on any proper subset $S$ of $\mathbb{Z}_n$ is invertible.* As shown from the Eisenstein irreducibility criterion and a change of variables trick, the so-called cyclotomic polynomial

$$\Phi_n(x) := p_{\mathbb{Z}_n}(x) = \sum_{k=0}^{n-1} x^k,$$

whose $n-1$ distinct complex roots are the $n-1$ nontrivial $n$th roots of unity, is irreducible over the field of rationals. Therefore if $P_S(\zeta_n^j)=0$ for some $j\in\{1,\dots,n-1\}$, then $P_S=\Phi_n$, i.e., $S=\mathbb{Z}_n$. Since $P_S(\zeta_n^0)=\#S>0$ for $S\neq\emptyset$, the result follows.

*Case* 2. *$n$ a prime power*. In general the $n$th cyclotomic polynomial $\Phi_n$ is defined as the (unique) monic polynomial of minimal degree satisfied by $\zeta_n=e^{2\pi i/n}$. When $n=p^\alpha$ with $p$ prime and $\alpha\geq 1$, $\Phi_n$ is an irreducible polynomial of degree $n(1-1/p)=(p-1)p^{\alpha-1}$ satisfied by the $(p-1)p^{\alpha-1}$ primitive $n$th roots of unity $\zeta_n^j$ with $p\nmid j$. It is well known that $\Phi_n$ is in this case

$$\Phi_n(x)=\sum_{k=0}^{p-1}x^{kp^{\alpha-1}}=\Phi_p(x^{p^{\alpha-1}}).$$

It is easy to see that a polynomial $P$ over the rationals $\mathbb{Q}$ of degree at most $n-1$ is satisfied by a primitive $n$th root of unity if and only if it is divisible by $\Phi_n$, which is in turn the case if and only if its sequence of coefficients is constant on each residue class of indices modulo $p^{\alpha-1}$.

Now suppose that $P(\zeta_n^j)=0$ for some (not necessarily primitive) $n$th root $\zeta_n^j$ with $j\neq 0$. Let $p^{\alpha-\beta}=(j,n)$ be the highest power of $p$ dividing $j$, so that $\zeta_n^j$ is a primitive $(p^\beta)$th root of unity. Define the polynomial $\tilde{P}$ from $P(x)=\sum_{k=0}^{n-1}a_kx^k$ by collecting all coefficients with indices in a common residue class modulo $p^\beta$:

$$\tilde{P}(x):=\sum_{k=0}^{p^\beta-1}\tilde{a}_kx^k,$$

$$\tilde{a}_k:=\tilde{a}(k;p^\beta):=\sum_{\ell=0}^{p^{\alpha-\beta}-1}a_{\ell p^\beta+k},\qquad k=0,\dots,p^\beta-1.$$

Then $\tilde{P}(\zeta_{p^\beta}^{j/(j,n)})=P(\zeta_n^j)=0$, and so the sequence $(\tilde{a}_k)$ is constant on each residue class for $k$ modulo $p^{\beta-1}$. Conversely, if for some $1\leq\beta\leq\alpha$ the reduced sequence $(\tilde{a}(k;p^\beta))$ has the specified property, then $P(\zeta_n^j)=0$ for all $j$ such that $(j,\ n)=p^{\alpha-\beta}$.

Applying these results to $P_S$ we see that *the Radon transform based on $S\neq\emptyset$ will fail to be invertible if and only if for some $\beta\in\{1,\dots,\alpha\}$ we have for each fixed $j\in\{0,\dots,p^{\beta-1}-1\}$ that the numbers*

$$\#(S\cap\{\ell p^\beta+kp^{\beta-1}+j:\ell\in\{0,\dots,p^{\alpha-\beta}-1\}\})$$

*are constant in $k\in\{0,\dots,p-1\}$*. More generally, if $B\subset\{1,\dots,\alpha\}$ is the set of all such $\beta$, then the nullity of the transform is $(1-1/p)\sum_{\beta\in B}p^\beta$, since there are $(1-1/p)p^\beta$ primitive $(p^\beta)$th roots of unity.

*Case* 3. *General $n$*. The facts about $\Phi_n$ are less simple in the "composite" case that $n$ has at least two distinct prime factors. In general, write $\phi(n)=n\prod_{\text{prime }p|n}(1-1/p)$ and

$$\mu(n)=\begin{cases}0 & \text{if } n \text{ is not square-free,}\\ (-1)^m & \text{if } n=p_1\dots p_m \text{ for } m\geq 0 \text{ distinct primes}\end{cases}$$

for the usual Euler and Möbius functions, respectively. It is well known that $\Phi_n$ is satisfied precisely by the $\phi(n)$ primitive $n$th roots of unity and has the expression

$$\Phi_n(x)=\prod_{d|n}(x^{n/d}-1)^{\mu(d)}. \tag{2.18}$$

In the composite case, unlike the prime power case, it is not necessarily true that all coefficients are 0 or 1.

Here is a (somewhat unsatisfying) criterion for invertibility of the Radon transform on $\mathbb{Z}_n$ based on $S$ for general $n$. *If for some divisor $d$ of $n$ the cyclotomic polynomial $\Phi_d$ divides $P_S$, then the transform is not one-to-one; otherwise it is invertible.* An algorithm of polynomial time in $n$ to decide the invertibility of the Radon transform based on translates of a given set $S$ can be constructed from this criterion.

We shall now present a rather powerful sufficient condition for $P_S$ to have a zero among the $n$th roots of unity, based on ideas from the prime power case. But we shall also demonstrate that the condition is not necessary.

Let $P(x) = \sum_{k=0}^{n-1} a_k x^k$. If the sequence $(a_0, \ldots, a_{n-1})$ is of the form

$$a_k = \sum_{\text{prime } p|n} \sum_{\ell=0}^{n/p\,-\,1} b_{p,\ell}\delta_{\ell,k \bmod \frac{n}{p}}, \tag{2.19}$$

then

$$P(\zeta_n) = \sum_{\text{prime } p|n} \sum_{\ell=0}^{n/p\,-\,1} b_{p,\ell} \sum_{j=0}^{p-1} \zeta_n^{jn/p\,+\,\ell} = 0.$$

More generally, let $d$ be a divisor on $n$. Collect all coefficients with indices in a common residue class modulo $d$:

$$\widetilde{P}_d(x) := \sum_{k=0}^{d-1} \tilde{a}(k;d)x^k,$$

$$\tilde{a}(k;d) := \sum_{\ell=0}^{n/d\,-\,1} a_{\ell d+k}, \qquad k = 0, \ldots, d-1. \tag{2.20}$$

If $\tilde{a}(\cdot;d)$ is a sequence of the form (2.19) (with $d$ replacing $n$), then

$$P(\zeta_n^{n/d}) = P(\zeta_d) = \widetilde{P}_d(\zeta_d) = 0.$$

Observe that in the prime power case $n = p^\alpha$, (2.19) reduces to the condition that the sequence $(a_k)$ be constant on each residue class of indices modulo $p^{\alpha-1}$; and the corresponding condition for $\tilde{a}(\cdot;d)$ reduces to the condition stated at the end of Case 2.

It is perhaps natural to conjecture that the converse is true: If $P(\zeta_n^j) = 0$ for some $j = 1, \ldots, n-1$, then for some divisor $d > 1$ of $n$ the sequence $\tilde{a}(\cdot;d)$ of (2.20) is of the form (2.19). But this conjecture is false, even when (as in our application to $P_S$) $(a_k)$ must consist of zeros and ones. For example, let $n = 30$, $S = \{0, 1, 7, 8, 18, 19, 20\}$, and let $P(x) = P_S(x) = \sum_{k \in S} x^k$. Then $P(1) = 7$ and it is easy to show that none of the coefficient sequences $\tilde{a}(\cdot;d)$ is of the form (2.19). Nevertheless,

$$P(x) = \Phi_{30}(x) \cdot (x^{12} + x^{10} + x^8 + x^7 + x^5 + x^3 + 1)$$

since by (2.18)

$$\Phi_{30}(x) = \frac{(x^{30}-1)(x^5-1)(x^3-1)(x^2-1)}{(x^{15}-1)(x^{10}-1)(x^6-1)(x-1)} = x^8 + x^7 - x^5 - x^4 - x^3 + x + 1;$$

thus $P(\zeta_{30}) = 0$.

**2.2. Reconstruction: least squares solutions.** Suppose that $g : \mathbb{Z}_n \to \mathbb{C}$, purported to be the Radon transform $g = \overline{f}$ of some $f : \mathbb{Z}_n \to \mathbb{C}$, is given and that reconstruction of $f$ from $\overline{f}$ is desired, but that $T$ is not invertible. In this section we shall utilize $U = T^{\div}$ to determine the extent to which such reconstruction is possible.

If $f \in \mathbb{C}^n$ is used as an approximate solution to

$$Tf = g, \tag{2.21}$$

then the error in the approximation can be measured by the squared length of the *residual vector* $R(f) = R := g - Tf$ in $\mathbb{C}^n$, namely, $\|R\|^2 = \|g - Tf\|^2 = \sum_{j=0}^{n-1} |g(j) - (Tf)(j)|^2$. If

$$f_0 := Ug \tag{2.22}$$

and $f = f_0 + h$ with $h \in \mathbb{C}^n$, then

$$\|R\|^2 = \|(I - TU)g - Th\|^2 = \|(I - TU)g\|^2 + \|Th\|^2, \tag{2.23}$$

the second equality following since $\mathcal{R}(T) = \mathcal{N}(I - TU) = \mathcal{R}(I - TU)^{\perp}$.

A *least squares solution* (LSS) to (2.21) is an $f \in \mathbb{C}^n$ minimizing $\|R\|^2$. From (2.23) it is evident that

$$f = f_0 + h \quad \text{is an LSS to (2.21) if and only if} \quad h \in \mathcal{N}(T), \tag{2.24}$$

and that any such choice will achieve the *least squares error* (LSE)

$$\text{LSE } = \|R_0\|^2 = \|f - f_0\|^2 = \|(I - TU)g\|^2. \tag{2.25}$$

If $h \in \mathcal{N}(T)$, the two pieces $f_0 = Ug$ and $h$ in the LSS $f = f_0 + h$ are orthogonal since, as we have seen, $\mathcal{R}(U) = \mathcal{N}(T)^{\perp}$, and so

$$\text{If} \quad f = f_0 + h \quad \text{is an LSS to (2.21), then} \quad \|f\|^2 = \|f_0\|^2 + \|h\|^2. \tag{2.26}$$

Thus (2.22) is the unique LSS of minimum norm.

Of course we know that

$$\text{The LSE (2.25) vanishes if and only if} \quad g \in \mathcal{R}(T), \tag{2.27}$$

a condition that can also be tested using (2.11).

*Reconstruction of probability mass functions.* We consider once again the problem of finding $f$ to minimize $\|R\|^2 = \|g - Tf\|^2$, but we add the probabilistically natural condition that $f$ be a probability mass function (pmf) on $\mathbb{Z}_n$, i.e., that

$$f(+) = 1, \tag{2.28a}$$

$$f(j) \geq 0 \quad \text{for every } j. \tag{2.28b}$$

Say that $f$ is an LSS$_1$ (respectively, an LSS$_{\geq 0}$) to (2.21) if $f$ satisfies (2.28a) (respectively, (2.28b)) and minimizes $\|R\|^2$ among all $f$ satisfying (2.28a) (respectively, (2.28b)). We shall begin by considering the two conditions separately. The condition (2.28a) is a simple one-dimensional linear constraint on $f \in \mathbb{C}^n$. Whereas $E = \{g - Tf : f \text{ satisfies (2.28b)}\}$ is a nonempty, closed convex set in $\mathbb{C}^n$, so that at least one LSS$_{\geq 0}$ exists, the analysis for (2.28b) is complicated by the facts that $E$ is not a linear subspace of $\mathbb{C}^n$ and that nonnegativity is not easily treated using Fourier techniques.

We begin with (2.28a). Let $\mathbf{1} \in \mathbb{C}^n$ denote the vector of ones. Any $f \in \mathbb{C}^n$ can be written in the form

$$f = f_0 + c\mathbf{1} + h$$

with $h(+) = 0$ and $nc = f(+) - f_0(+) = f(+) - n^{1/2}\widehat{f_0}(0) = f(+) - g(+)/\#S$. Then by (2.23)

$$\begin{aligned}\|R\|^2 &= \|R_0\|^2 + \|T(c\mathbf{1} + h)\|^2 = \|R_0\|^2 + \|c(\#S)\mathbf{1} + Th\|^2 \\ &= \|R_0\|^2 + c^2 n(\#S)^2 + \|Th\|^2.\end{aligned}$$

The constraint (2.28a) amounts to $nc = 1 - g(+)/\#S$, whence

$$\|R\|^2 = \|R_0\|^2 + n^{-1}[\#S - g(+)]^2 + \|Th\|^2. \tag{2.29}$$

It is now clear that

(2.30) $\quad f$ is an LSS$_1$ to (2.21) if and only if $f = f_1 + h$ with $h \in \mathcal{N}(T)$,

where

$$f_1 := f_0 + n^{-1}[1 - g(+)/\#S], \tag{2.31}$$

and that any such choice will achieve the minimum error

$$\text{LSE}_1 = \|R_0\|^2 + n^{-1}[\#S - g(+)]^2. \tag{2.32}$$

An easy calculation shows that

(2.33) $\quad$ If $f = f_1 + h$ is an LSS$_1$ to (2.21), then $\|f\|^2 = \|f_1\|^2 + \|h\|^2$,

so that (2.31) is the unique LSS$_1$ of minimum norm, and that

$$\|f_1\|^2 = \|f_0\|^2 + n^{-1}[1 - (g(+)/\#S)^2]. \tag{2.34}$$

We have found in particular that

(2.35) $\quad$ The four conditions
LSE$_1$ = LSE, $\quad f_1 = f_0, \quad f_0(+) = 1,\quad$ and $g(+) = \#S$
are equivalent.

As for the more difficult condition (2.28b), we shall focus on the question of when LSE$_{\geq 0}$ equals LSE. According to (2.24), this equality will hold if and only if $f_0 + h \geq 0$ for some $h \in \mathcal{N}(T)$, in which case all such $f = f_0 + h$ make up the set of LSS$_{\geq 0}$. Even the minimum $\|R\|^2$ for $f$ satisfying both the conditions in (2.28) will equal (2.25) if we suppose further that $f_0(+) = 1$. Having nothing more to say in general, we pass to our two examples, in which we suppose that the given $g$, and hence $f_0$, is real valued.

*Example* 2.4. (a) *Shells*. Suppose that $T$ is not invertible. Define

$$m_j := \min\left\{ f_0(j + \ell \cdot 4(n,r)) : 0 \le \ell < \frac{n}{4(n,r)} \right\}, \qquad 0 \le j < 4(n,r). \tag{2.36}$$

If $f = f_0 + h \ge 0$ with $h \in \mathcal{N}(T)$, then $m_j + h(j) \ge 0$ and $m_{j+2(n,r)} - h(j) \ge 0$ for $0 \le j < 2(n,r)$, and so

$$m_j + m_{j+2(n,r)} \ge 0 \quad \text{for} \quad 0 \le j < 2(n,r). \tag{2.37}$$

Conversely, if (2.37) holds and $h \in \mathcal{N}(T)$ satisfies

$$-m_j \le h(j) \le m_{j+2(n,r)}, \qquad 0 \le j < 2(n,r), \tag{2.38}$$

(clearly such $h$'s exist), then $f = f_0 + h \ge 0$.

In particular, we have shown that

$$\mathrm{LSE}_{\ge 0} = \mathrm{LSE} \quad \text{if and only if the numbers} \quad m_j \quad \text{of (2.36) satisfy (2.37).} \tag{2.39}$$

Combining (2.27), (2.14), (2.35), and (2.39), we have the following summary: $g \in \mathbb{R}^n$ *is the Radon transform of some probability mass function* $f$ *if and only if the conditions* (2.14), $g(+) = \#S$, *and* (2.37) *hold, in which case the most general such* $f$ *is of the form*

$$f = f_0 + h,$$

*where* $f_0 = Ug$ *is defined by* (2.15) *and* $h : \mathbb{Z}_n \to \mathbb{R}$ *is any function satisfying* (2.38) *and*

$$\begin{aligned} h(j + 2(n,r)) &= -h(j), && 0 \le j < 2(n,r), \\ h(j + \ell \cdot 4(n,r)) &= h(j), && 0 \le \ell < \frac{n}{4(n,r)}. \end{aligned}$$

For general $g \in \mathbb{C}^n$ we have, from the form of $U$ computed in §5.1 below (see (5.3)),

$$\mathrm{LSE} = \frac{2(n,r)}{n} \sum_{j=0}^{2(n,r)-1} \left| \sum_{\ell=0}^{n/[2(n,r)]\,-\,1} (-1)^\ell g(j + \ell \cdot 2(n,r)) \right|^2. \tag{2.40}$$

(b) *Balls*. Define

$$m_j := \min\{f_0(j + \ell \cdot (n,\nu)) : 0 \le \ell < n/(n,\nu)\}, \qquad 0 \le j < (n,\nu). \tag{2.41}$$

If $f = f_0 + h \ge 0$ with $h \in \mathcal{N}(T)$, then $m_j + h(j) \ge 0$ for $0 \le j < (n,\nu)$ and $h(+) = 0$, so

$$m_+ := \sum_{j=0}^{(n,\nu)-1} m_j \ge 0. \tag{2.42}$$

Conversely, if (2.42) holds and $h \in \mathcal{N}(T)$ is defined by

$$h(j + \ell \cdot (n,\nu)) = \epsilon_j - m_j, \quad 0 \le j < (n,\nu), \quad 0 \le \ell < \frac{n}{(n,\nu)}, \tag{2.43}$$

where

$$\epsilon_j \geq 0 \quad \text{for every } j \text{ and} \quad \epsilon_+ := \sum_{j=0}^{(n,\nu)-1} \epsilon_j = m_+, \tag{2.44}$$

then $f = f_0 + h \geq 0$.

Therefore

$$\text{LSE}_{\geq 0} = \text{LSE} \quad \text{if and only if the numbers } m_j \text{ of (2.41) satisfy (2.42),} \tag{2.45}$$

and we have the following summary: *$g \in \mathbb{R}^n$ is the Radon transform of some probability mass function $f$ if and only if the conditions*

$$\sum_{k \equiv j \pmod{(n,\nu)}} g(k) = \frac{(\#S)}{(n,\nu)}, \qquad 0 \leq j < (n,\nu), \tag{2.46}$$

*and* (2.42) *hold, in which case the most general such $f$ is of the form*

$$f = f_0 + h,$$

*where $f_0 = Ug$ is defined by* (2.17) *and $h : \mathbb{Z}_n \to \mathbb{R}$ is any function satisfying* (2.43)-(2.44).

For general $g \in \mathbb{C}^n$ we have, from the form of $U$ computed in Section 5.2 below (see (5.10)),

$$\text{LSE} = \frac{(n,\nu)}{n} \sum_{j=0}^{(n,\nu)-1} \left| \sum_{k \equiv j \pmod{(n,\nu)}} g(k) - \frac{g(+)}{(n,\nu)} \right|^2. \tag{2.47}$$

**3. Nearly all Radon transforms based on translates of large sets are invertible.** Define $\mathcal{S}_t := \{S \subset \mathbb{Z}_n : \#S = t\}$. Let $S \in \mathcal{S}_t$. As shown in Section 2.1 above, the Radon transform based on translates of $S$ is invertible if and only if $\sum_{k \in S} e^{2\pi i k/d} = 0$ for some divisor $d$ of $n$. For large $t$, Diaconis and Graham [4] conjectured that the Radon transform based on translates of $S$ is invertible for most sets $S \in \mathcal{S}_t$ when $t$ (and hence $n$) is suitably large. This conjecture is confirmed by the following precise result, which is the heart of the present section.

THEOREM 3.1. *Let $S$ be a random subset of size $t$ from $\mathbb{Z}_n$, and let $p(n,t)$ denote the probability that the Radon transform based on $S$ fails to be invertible. If $(t_n)$ is a sequence of positive integers with* $\min(t_n, n - t_n) \to \infty$ *as $n \to \infty$, then we have the uniform convergence*

$$\sup\{p(n,t) : t_n \leq t \leq n - t_n\} \to 0 \quad \text{as } n \to \infty. \tag{3.1}$$

Observe that $\hat{\chi}_S + \hat{\chi}_{S^c} = \hat{\chi}_{\mathbb{Z}_n}$ vanishes except at the origin. Hence if $S$ is any nonempty proper subset of $\mathbb{Z}_n$, then the Radon transforms based on $S$ and on $S^c$ have the same nullity. We therefore may and henceforth do suppose that $t_n \leq \lfloor n/2 \rfloor$ and restrict the range of $t$ in (3.1) to $t_n \leq t \leq \lfloor n/2 \rfloor$. We continue to suppose $t_n \to \infty$.

Theorem 3.1, with its conclusion (3.1) weakened to $p(n, t_n) \to 0$, can be related to Theorem 1 in Freedman and Lane [8], which concerns the asymptotic empirical distribution of the Fourier coefficients of a random permutation of a fixed vector in $\mathbb{C}^n$. We begin by observing that the sequence $\chi_S(\cdot)$, indexed by $\mathbb{Z}_n$, is a random permutation of the vector consisting of $t \equiv t_n$ copies of one followed by $n-t$ copies of zero. Define the vector $z^n$ to consist of $t$ copies of $+\sqrt{(n-t)/t}$ followed by $n-t$ copies of $-\sqrt{t/(n-t)}$; this is the normalization appropriate to meet condition (i) of Theorem 1 in Freedman and Lane. Condition (ii) also holds, so the theorem implies that the random measure $\mu_n$ placing mass $1/n$ at each of the $n$ complex values zero and

$$(\sqrt{(n-t)/t} + \sqrt{t/(n-t)}\,)\,\hat{\chi}_S(j), \qquad j = 1, \dots, n-1,$$

converges in probability to the standard complex normal distribution, i.e., to the distribution of $U + iV$, where $U$ and $V$ are independent real normal variables with mean zero and variance $\frac{1}{2}$. In particular, the nullity of the Radon transform based on $S$ equals $n\mu_n\{0\} - 1$ and so is $o(n)$ in probability, but this falls far short of the desired assertion that $P\{\text{nullity } = 0\} \to 1$.

Nevertheless, the preparatory lemma of Freedman and Lane [8, Lemma 4] *can* be used to establish Theorem 3.1 in its full strength. A key ingredient in our proof below of Theorem 3.1 is the following estimate (cf. (3.4) below):

$$P\left\{\sum_{k\in S} e^{2\pi i k/d} = 0\right\} = O(t_n^{-1/2}) \tag{3.2}$$

uniformly for divisors $d > 1$ of $n$ and $t_n \le t \le \lfloor n/2 \rfloor$. In our proof (3.2) will be established effortlessly using bounds on the coarseness of random sums due to Fill [5], but here we sketch an alternative proof of (3.2). Apply the aforementioned Lemma 4 taking $x = (t$ copies of $+\sqrt{(n-t)/t}$, $n-t$ copies of $-\sqrt{t/(n-t)}\,)$, $y = 0$, and, for $j = 1, \dots, n$, $a_j = \sqrt{2}\,\cos(2\pi j/d)$ and $b_j = \sqrt{2}\,\sin(2\pi j/d)$ where $d > 1$ is a fixed divisor of $n$. From the proof of the lemma, which uses bounds produced by the deep analysis of Ho and Chen [9] for a combinatorial central limit theorem dating back to Wald and Wolfowitz [11], we find

$$\sup_{u\in\mathbf{R}} |\widetilde{F}(u) - \Phi(u)| \le 48\sqrt{2}\, n^{-1/2}\,\sqrt{(n-t)/t},$$

where (in the notation of Freedman and Lane) $\widetilde{F}$ is the distribution function of $\widetilde{W} = (1 - 1/n)^{1/2} \cdot \sqrt{2}\,n^{-1/2}\,\mathrm{Re}\big(\sum_{k\in S} e^{2\pi i k/d}\big)$ and $\Phi$ is the standard real normal distribution function. In particular,

$$P\left\{\sum_{k\in S} e^{2\pi i k/d} = 0\right\} \le 96\sqrt{2}\,n^{-1/2}\sqrt{(n-t)/t} \le 96\sqrt{2}\,t_n^{-1/2}, \tag{3.3}$$

independently of $d$, and (3.2) is established.

*Proof of Theorem* 3.1. The first step is subadditivity:

$$\begin{aligned} p(n,t) &= P\left\{\sum_{k\in S} e^{2\pi i k/d} = 0 \quad \text{for some } d|n\right\} \\ &\le \sum_{d|n} p(d; n, t) \end{aligned}$$

where

$$p(d; n, t) := P\left\{\sum_{k\in S} e^{2\pi i k/d} = 0\right\}.$$

Of course $p(1; n, t) = 0$. For the remaining divisors $d$, note that $\sum_{k \in S} e^{2\pi i k/d}$ is the sum of a random sample of size $t$ without replacement from the population $\{e^{2\pi i k/d} : k \in \mathbb{Z}_n\}$ having $n/d$ copies each of $e^{2\pi i k/d}$, $k \in \mathbb{Z}_d$. From Corollary 2.9 if $n$ is even and Corollary 2.14 if $n$ is odd in Fill [5] (see also Remark 2.10 therein), it follows that $p(d; n, t)$ is no bigger than the modal probability of the hypergeometric ($\lfloor n/2 \rfloor$, $\lceil n/2 \rceil$; $t$) distribution, namely,

$$\binom{\lfloor n/2 \rfloor}{\lfloor t/2 \rfloor}\binom{\lceil n/2 \rceil}{\lceil t/2 \rceil}\Big/\binom{n}{t} = \binom{t}{\lfloor t/2 \rfloor}\binom{n-t}{\lfloor (n-t)/2 \rfloor}\Big/\binom{n}{\lfloor n/2 \rfloor}.$$

From here the bound

$$p(d; n, t) \le (1 + o(1)) \frac{2}{\sqrt{\pi}} t_n^{-1/2} = O(t_n^{-1/2}), \tag{3.4}$$

independent of $d > 1$ dividing $n$ and uniform for $t_n \le t \le \lfloor n/2 \rfloor$, follows simply. In passing we note that the constant factor $2/\sqrt{\pi}$ provided by the simple coarseness results of Fill [5] is considerably smaller than the corresponding constant $96\sqrt{2}$ appearing in (3.3).

So we find

$$p(n, t) = O\left(\tau(n)/\sqrt{t_n}\right) \quad \text{uniformly for} \quad t_n \le t \le \lfloor n/2 \rfloor,$$

where $\tau(n)$ is the number of positive divisors of $n$ and grows quite slowly with $n$; it is easy to see that $\tau(n) = o(n^\epsilon)$ for any $\epsilon > 0$. If $t_n \gg \tau^2(n)$, the proof of Theorem 3.1 is complete. To complete the proof of Theorem 3.1 in general it suffices to assume $t_n \le \lfloor n^{1/3} \rfloor$ and show

$$p(n, t) = o(1) \quad \text{uniformly for} \quad t_n \le t \le \lfloor n^{1/3} \rfloor. \tag{3.5}$$

It follows from (3.4) in general that

$$p(n, t) \le (1 + o(1)) \frac{2}{\sqrt{\pi}} t_n^{-1/2} \#\{d \le M : d | n\} + \sum_{d : M < d | n} p(d; n, t) \tag{3.6}$$

uniformly for $t_n \le t \le \lfloor n/2 \rfloor$ for any positive integer $M$. The first term on the right in (3.6) is no bigger than $(1 + o(1))M\,(2/\sqrt{\pi})t_n^{-1/2} = o(1)$ for any fixed $M$, so we turn next to an analysis of the second term.

If $U = U(d; n, t)$ is the sum of the second coordinates of a random sample of size $t$ *with* replacement from the population $\{(k, e^{2\pi i k/d}) : k = 0, \dots, n-1\}$, then $U$, independently of $n$, has the same distribution as any sum $U(d; t)$ of $t$ independent random variables each uniformly distributed on the $d$ $d$th roots of unity, and

$$\begin{aligned} P\{U = 0\} &\ge P\{\text{all sampled items distinct}\}P\{U = 0 | \text{all items distinct}\} \\ &= (1 - 1/n)(1 - 2/n) \dots (1 - (t-1)/n)p(d; n, t), \end{aligned}$$

i.e.,

$$p(d; n, t) \le [(1 - 1/n)(1 - 2/n) \dots (1 - (t-1)/n)]^{-1} P\{U = 0\}. \tag{3.7}$$

We estimate $P\{U = 0\} = P\{U(d;t) = 0\}$ very crudely. If $d > 1$ is odd, it is simple to show that the modal probability for $U(d;2)$ equals $2/d^2$. If $d$ is even, then $P\{U(d;2) = 0\} = 1/d$, but the modal probability for $U(d;3)$ equals $3(d-1)/d^3 \leq 3/d^2$.

Since convolution can only decrease modal probabilities, we find

$$P\{U(d;t) = 0\} \leq 3/d^2 \quad \text{for all } d > 1 \text{ and } t \geq 3. \tag{3.8}$$

Combining (3.6)–(3.8) we find

$$\begin{aligned} p(n,t) &\leq (1+o(1))M\frac{2}{\sqrt{\pi}}\, t_n^{-1/2} + 3\left(1 - \frac{n^{1/3}}{n}\right)^{-n^{1/3}} \sum_{d:M<d|n} d^{-2} \\ &\leq (1+o(1))\; 3 \sum_{d>M} d^{-2} \end{aligned}$$

uniformly for $t_n \leq t \leq \lfloor n^{1/3} \rfloor$ for each $M$. As $M$ is arbitrary, (3.5) holds and the proof is finished.

## 4. Inversion formulas for shells and balls of arbitrary radius.

In this section we present useful inversion formulas (Theorems 4.2 and 4.5) for Radon transforms based on shells $S_r = \{-r, r\}$ and balls $S_r^+ = \{-r, \ldots, r\}$ of arbitrary radius $r$ satisfying $2r + 1 \leq n$. These results are displayed numerically for $r = 1, 2$ in Tables 4.1–4.4.

Criteria for invertibility in the two cases were given in Example 2.3 above; we shall investigate only invertible transforms in this section. In §2 above we used Fourier analysis to write down an inversion formula valid for any invertible Radon transform on $\mathbb{Z}_n$, but the particular results developed in the present section for shells and balls will be seen in §4.3 below to be of an appreciably simpler nature.

In §5 below we shall extend our analysis to noninvertible transforms.

### 4.1. Shells.

According to the following theorem, the inverse transform can be found by solving a simple (circular) difference equation.

THEOREM 4.1. *Let $2r+1 \leq n$ and suppose that $n$ is not a multiple of $2^{\alpha+2}$, where $2^\alpha$ is the largest power of two that divides $r$. Then there exists a unique function $u : \mathbb{Z}_n \to \mathbb{C}$ satisfying*

$$u(k-r) + u(k+r) = \begin{cases} 1 & \text{if } k = 0 \text{ (in } \mathbb{Z}_n\text{)}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

*Moreover, if, given $f : \mathbb{Z}_n \to \mathbb{C}$, $\bar{f} : \mathbb{Z}_n \to \mathbb{C}$ is its Radon transform*

$$\bar{f}(k) = f(k-r) + f(k+r), \tag{4.2}$$

*then*

$$f(j) = \sum_{k \in \mathbb{Z}_n} u(k)\bar{f}(j+k). \tag{4.3}$$

*Proof.* Under the stated conditions we know that the transform is invertible, so it is possible to write

$$f(0) = \sum_{k \in \mathbb{Z}_n} u(k)\bar{f}(k)$$

identically in $f$ for some $u$. By matching coefficients of values of $f$ we find that $u$ satisfies (4.1). Conversely, if (4.1) is satisfied by a given $u$, then so is (4.3). Since the Radon transform is linear and invertible, a trivial vector space dimension argument establishes uniqueness of $u$.

The solution $u$ to (4.1) is not hard to find. First note that $u$ is symmetric. Then by setting $k = 2\ell r$ for successive values of $\ell$ we find

$$u((2\ell+1)r) = (-1)^\ell \frac{1}{2}, \qquad 0 \le \ell < \ell_0, \tag{4.4}$$

where $\ell_0$ is the smallest positive integer $\ell$ such that $2\ell r \equiv 0 \pmod{n}$. We can check that $\ell_0$ is the odd number

$$\ell_0 = \begin{cases} n/[2(n,r)] & \text{if } n \text{ is a (necessarily odd) multiple of } 2^{\alpha+1}, \\ n/(n,r) & \text{otherwise.} \end{cases} \tag{4.5}$$

Conversely, if $u$ is defined so as to vanish identically except for the stipulation that (4.4) should hold, then the resulting $u$ does indeed satisfy (4.1).

It is evident that $u(j) = u(j+4r)$ is an identity in $0 \le j < j+4r < n$. It follows that for $0 \le j < n$, $u(j)$ has constant value on each residue class for $j$ modulo $4r$. It is also not hard to see that if an inverse $u_0$ exists on $\mathbb{Z}_n$, then an inverse $u$ exists on $\mathbb{Z}_{n+k(4r)}$ for any integer $k \ge 0$; to wit, $u(j) = u_0(j \bmod 4r)$ when $0 \le j < n + k(4r)$. Thus, for example, in constructing tables for $u(j)$ on $\mathbb{Z}_n$, $0 \le j < n$, we need only concern ourselves with the values of $n \bmod 4r$ and $j \bmod 4r$.

THEOREM 4.2. *The unique solution $u$ to* (4.1) *is obtained by letting* (4.4)–(4.5) *hold and setting all remaining values of $u$ to zero. The resulting $u$ provides the inverse to the Radon transform* (4.2), *in the sense that* (4.3) *holds.*

*Example* 4.3. For the cases $r = 1, 2$, Tables 4.1 and 4.2 give the values of $u(j)$ when $0 \le j < n$.

TABLE 4.1
*Inverse Radon transform $u$ for shells of radius $r = 1$.*

| | | $j \bmod 4$ | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 |
| | 0 | no inverse | | | |
| | 1 | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ |
| $n \bmod 4$ | 2 | 0 | $+\frac{1}{2}$ | 0 | $-\frac{1}{2}$ |
| | 3 | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ |

TABLE 4.2
*Inverse Radon transform*
*$u$ for shells of radius $r = 2$.*

| | | $j$ mod 8 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| $n$ mod 8 | 0 | no inverse | | | | | | | |
| | 1 | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ |
| | 2 | $+\frac{1}{2}$ | 0 | $+\frac{1}{2}$ | 0 | $-\frac{1}{2}$ | 0 | $-\frac{1}{2}$ | 0 |
| | 3 | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ |
| | 4 | 0 | 0 | $+\frac{1}{2}$ | 0 | 0 | 0 | $-\frac{1}{2}$ | 0 |
| | 5 | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ |
| | 6 | $-\frac{1}{2}$ | 0 | $+\frac{1}{2}$ | 0 | $+\frac{1}{2}$ | 0 | $-\frac{1}{2}$ | 0 |
| | 7 | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $+\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ |

**4.2. Balls.** The next result sets up the difference equation analogous to (4.1) for a ball of radius $r$.

THEOREM 4.4. *Let $2r+1 \le n$ and suppose that $(n, 2r+1) = 1$. Then there exists a unique function $u : \mathbb{Z}_n \to \mathbb{C}$ satisfying*

$$u(k-r) + \ldots + u(k+r) = \begin{cases} 1, & \text{if } k = 0 \text{ (in } \mathbb{Z}_n\text{)}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

*Moreover, if, given $f : \mathbb{Z}_n \to \mathbb{C}$, $\bar{f} : \mathbb{Z}_n \to \mathbb{C}$ is its Radon transform*

$$\bar{f}(k) = f(k-r) + \ldots + f(k+r), \tag{4.7}$$

*then*

$$f(j) = \sum_{k \in \mathbb{Z}_n} u(k)\bar{f}(j+k). \tag{4.8}$$

Next we show how to solve (4.6) for the (symmetric) function $u$. The results are presented in Theorem 4.5 below. For simplicity we employ the notation $\nu = 2r+1$ for the number of values summed in forming the transform (4.7) and $j^*$ for that value in the set $\{-r+1, \ldots, r+1\}$ of $\nu$ elements to which a given integer $j$ is congruent modulo $\nu$. That is, $j^* = (j + r - 1) \bmod \nu \; - \; (r-1)$. We begin by subtracting successive instances of (4.6) to produce the simpler identity

$$u(j) - u(j-\nu) = \begin{cases} +1 & \text{if } j = r \text{ (in } \mathbb{Z}_n\text{)}, \\ -1 & \text{if } j = r+1 \text{ (in } \mathbb{Z}_n\text{)}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

Using (4.9), we find it simple to derive

$$u(-r) = u(n-r) = u((n-r)^*).$$

The equality

$$u((n-r)^*) = u(n+(n-r)^*) = u((2n-r)^*)$$

follows in the same manner, provided $(n-r)^*$ is congruent to neither $r$ nor $r+1$ modulo $n$, that is, provided $n \not\equiv -1 \pmod{\nu}$ and $n \not\equiv 0 \pmod{\nu}$. Likewise,

$$u((2n-r)^*) = u((3n-r)^*)$$

provided $2n \not\equiv -1 \pmod{\nu}$ and $2n \not\equiv 0 \pmod{\nu}$. Continuing along these lines, we find

$$(4.10) \qquad u(-r) = u((\ell n - r)^*), \qquad 1 \le \ell \le \ell_-,$$

where $\ell_-$ is the smallest positive integer $\ell$ such that $\ell n \equiv -1 \pmod{\nu}$. (Note that since by assumption $(n,\nu)=1$, $\ell n \not\equiv 0 \pmod{\nu}$ for $1 \le \ell < \nu$, while $\ell n \equiv -1 \pmod{\nu}$ for some such value, namely, $\ell_-$.) As (4.10) is obtained, so is

$$(4.11) \qquad u(-r-1) = u((\ell n + r)^*), \qquad 1 \le \ell \le \ell_+,$$

where $\ell_+$ is the smallest positive integer $\ell$ such that $\ell n \equiv +1 \pmod{\nu}$.

It is easy to show that $\ell_- + \ell_+ = \nu$ and that the $\nu$ $u$-indices appearing on the right in (4.10)–(4.11) are distinct and make up the set $\{-r+1,\dots,r+1\}$. Thus from (4.6) with $k=1$ we see

$$\ell_- u(-r) + \ell_+ u(-r-1) = 0.$$

On the other hand, from (4.9) with $j=r$ we find

$$u(-r) - u(-r-1) = 1$$

using the symmetry of $u$. Hence

$$(4.12) \qquad u(-r) = \ell_+/\nu, \qquad u(-r-1) = -\ell_-/\nu.$$

As with shells, in constructing tables for $u(j)$ on $\mathbb{Z}_n$, $0 \le j < n$, we need only concern ourselves with the values of $n$ mod $\nu$ and $j$ mod $\nu$.

THEOREM 4.5. *The unique solution $u$ to* (4.6) *has only two values, namely $\ell_+/\nu$ and $-\ \ell_-/\nu$, where $\nu = 2r+1$ and $\ell_+ = \nu - \ell_-$ is defined to be the smallest positive integer $\ell$ such that $\ell n \equiv +1 \pmod{\nu}$. The solution is obtained by first defining $u(-r)$ and $u(-r-1)$ by* (4.12); *next using* (4.10)–(4.11), *where $j \equiv j^* \in \{-r+1,\dots,r+1\} \pmod{\nu}$, to set the values of $u(-r+1),\dots,u(r+1)$; and finally using* (4.9) *to set the remaining values of $u$. The resulting $u$ provides the inverse to the Radon transform* (4.7), *in the sense that* (4.8) *holds.*

*Example* 4.6. For the cases $r = 1,2$, Tables 4.3 and 4.4 give the values of $u(j)$ when $0 \le j < n$.

TABLE 4.3
*Inverse Radon transform $u$ for balls of radius $r=1$.*

| | | $j$ mod 3 | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| | 0 | no inverse | | |
| $n$ mod 3 | 1 | $+\frac{1}{3}$ | $+\frac{1}{3}$ | $-\frac{2}{3}$ |
| | 2 | $-\frac{1}{3}$ | $+\frac{2}{3}$ | $-\frac{1}{3}$ |

TABLE 4.4
*Inverse Radon transform*
*$u$ for balls of radius $r = 2$.*

| | | $j$ mod 5 | | | | |
|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 |
| $n$ mod 5 | 0 | no inverse | | | | |
| | 1 | $+\frac{1}{5}$ | $+\frac{1}{5}$ | $+\frac{1}{5}$ | $-\frac{4}{5}$ | $+\frac{1}{5}$ |
| | 2 | $+\frac{3}{5}$ | $-\frac{2}{5}$ | $+\frac{3}{5}$ | $-\frac{2}{5}$ | $-\frac{2}{5}$ |
| | 3 | $-\frac{3}{5}$ | $+\frac{2}{5}$ | $+\frac{2}{5}$ | $-\frac{3}{5}$ | $+\frac{2}{5}$ |
| | 4 | $-\frac{1}{5}$ | $-\frac{1}{5}$ | $+\frac{4}{5}$ | $-\frac{1}{5}$ | $-\frac{1}{5}$ |

## 4.3. Comparison of inversion formulas.

**4.3. Comparison of inversion formulas.** Suppose that the Radon transform on $\mathbb{Z}_n$ based on the shell $S_r$ is invertible, and let $u$ be its inverse, in the sense of (4.3). It follows from the Fourier calculation (2.15) that

$$u(k) = \frac{1}{2n} \sum_{j=0}^{n-1} \frac{\cos(2\pi jk/n)}{\cos(2\pi jr/n)}, \qquad k \in \mathbb{Z}_n. \tag{4.13}$$

By Theorem 4.2, these $u$'s simplify to (4.4)–(4.5) for $k$ congruent to an odd multiple of $r$ modulo $n$ and vanish otherwise. An identity for the trigonometric sum on the right in (4.13) is obtained as a byproduct of the comparison. Clearly Theorem 4.2 is a great practical improvement on (4.13).

Likewise, for balls Theorem 4.5 improves on the Fourier result

$$u(k) = \frac{1}{n}\left[\frac{1}{\nu} + \sum_{j=1}^{n-1} \frac{\sin(\pi j/n)}{\sin(\pi j\nu/n)} \cos\left(\frac{2\pi jk}{n}\right)\right], \qquad k \in \mathbb{Z}_n. \tag{4.14}$$

## 5. Generalized inverses for shells and balls of arbitrary radius.

**5. Generalized inverses for shells and balls of arbitrary radius.** In this section we continue our non-Fourier investigation of Radon transforms on $\mathbb{Z}_n$ based on translates of shells $S_r = \{-r, r\}$ and balls $S_r^+ = \{-r, \ldots, r\}$ of radius $r$ with $2r+1 \le n$ begun in the previous section. Here we consider only transforms that are *not* invertible.

**5.1. Shells.** Throughout §5.1 we suppose that $2r+1 \le n$ and that $n$ is a multiple of $2^{\alpha+2}$, where $2^\alpha$ is the largest power of two that divides $r$. That is, we suppose that the Radon transform $T$ on $\mathbb{Z}_n$ based on the shell $S_r$ is *not* invertible.

A simpler form of the M-P inverse (2.15) is provided by our first result.

THEOREM 5.1. *If $u : \mathbb{Z}_n \to \mathbb{R}$ is defined by setting*

$$u((2\ell+1)r) := (-1)^\ell \frac{1}{2}\left[1 - (2\ell + 1)\frac{2(n,r)}{n}\right], \quad 0 \le \ell < \frac{n}{2(n,r)}, \tag{5.1}$$

*and $u(j) := 0$ otherwise, then $u$ is symmetric and the transformation*

$$(Ug)(j) = \sum_{k \in \mathbb{Z}_n} u(k) g(j+k) \qquad (g : \mathbb{Z}_n \to \mathbb{R}), \tag{5.2}$$

*i.e., the matrix* $\operatorname{circ}(u(-\cdot)) = \operatorname{circ}(u)$, *provides the* M-P *inverse to the Radon transform*

$$(Tf)(k) = \bar{f}(k) = f(k-r) + f(k+r)$$

*based on $S_r$.*

*Proof.* For any $u : \mathbb{Z}_n \to \mathbb{C}$ the corresponding mapping $U$ of (5.2) satisfies $TUT = T$ if and only if $TUTf = Tf$ for every coordinate vector $f$, i.e., if and only if $u$ satisfies

$$u(k-2r) + 2u(k) + u(k+2r) = \begin{cases} 1 & \text{if } k = \pm r \text{ (in } \mathbb{Z}_n), \\ 0 & \text{otherwise.} \end{cases}$$

Using the observation that $n/[2(n,r)]$ is even and the smallest positive integer $\ell$ for which $2\ell r \equiv 0 \pmod n$, we can show by direct calculation that the function $u$ of (5.1) is indeed a solution to this difference equation. Since $T$ and the resulting $U$ are circulants, they commute. Finally, one can check directly that $UTU = U$.

By direct calculation, the image of the usual zeroth coordinate vector $e_0$ in $\mathbb{R}^n$ under $I - TU$ is the multiple $2(n,r)/n$ of the vector $w \in \mathbb{R}^n$ defined by

$$w(k) = \begin{cases} (-1)^\ell & \text{if } k \equiv 2\ell r \pmod n \text{ for some } \ell, \\ 0 & \text{otherwise,} \end{cases}$$

$$= \begin{cases} (-1)^\ell & \text{if } k \equiv \ell \cdot 2(n,r) \pmod n \text{ for some } 0 \le \ell < n/[2(n,r)], \\ 0, & \text{otherwise.} \end{cases}$$

Hence for general $g \in \mathbb{C}^n$

$$\text{(5.3)} \qquad ((I - TU)g)(j) = \frac{2(n,r)}{n} \sum_{\ell=0}^{n/[2(n,r)]\, -\, 1} (-1)^\ell g(j + \ell \cdot 2(n,r)), \qquad j \in \mathbb{Z}_n,$$

and (2.40) above follows.

From the equality of (5.2) and (2.15) we have the byproduct

$$\text{(5.4)} \qquad u(k) = \frac{1}{2n} \sum_{j=0}^{n-1} \left[\cos\left(\frac{2\pi jr}{n}\right)\right]^{\div} \cos\left(\frac{2\pi jk}{n}\right), \qquad k \in \mathbb{Z}_n,$$

for the sequence $u$ described in Theorem 5.1. This result is a companion to (4.13) above.

**5.2. Balls.** Throughout §5.2 we suppose that $\nu = 2r + 1 \le n$ and that $(n, \nu) > 1$, so that the Radon transform $T$ on $\mathbb{Z}_n$ based on the ball $S_r^+$ is *not* invertible.

We begin with an analogue of Theorem 5.1 for balls that display the M-P inverse (2.17) in simplified form.

THEOREM 5.2. *Define $u : \mathbb{Z}_n \to \mathbb{R}$ in three stages. First, set*

$$\text{(5.5)} \qquad u(\ell\nu - r) = \frac{1}{n\nu} + \frac{(n,\nu)}{n}\left(\ell - \frac{1}{2}\right) - \frac{1}{2}, \qquad 0 < \ell \le \frac{n}{(n,\nu)};$$

*then define $u(-(\ell\nu - r)) := u(\ell\nu - r)$ for the same values of $\ell$; and finally set*

$$\text{(5.6)} \qquad u(j) := \frac{1}{n\nu}$$

*for the remaining values of $j$. Then $u$ is symmetric and*

$$\text{(5.7)} \qquad (Ug)(j) = \sum_{k \in \mathbb{Z}_n} u(k) g(j+k) \qquad (g : \mathbb{Z}_n \to \mathbb{R}),$$

*i.e., the matrix* $\operatorname{circ}(u(-\cdot)) = \operatorname{circ}(u)$, *provides the* M-P *inverse to the Radon transform*

$$(Tf)(k) = \bar{f}(k) = f(k-r) + \ldots + f(k+r)$$

*based on* $S_r^+$.

*Proof.* The mapping $U$ of (5.7) satisfies $TUT = T$ if and only if $u$ satisfies

$$\sum_{\ell=-r}^{r} \sum_{m=-r}^{r} u(k+\ell+m) = \begin{cases} 1 & \text{if } k \in \{-r,\ldots,r\} \text{ (in } \mathbb{Z}_n), \\ 0 & \text{otherwise,} \end{cases}$$

i.e., $T^2 u = \chi_{\{-r,\ldots,r\}}$. Now define $u$ as in the theorem and $v : \mathbb{Z}_n \to \mathbb{R}$ by

$$\begin{aligned} v(0) &= 1 - \frac{[(n,\nu)-1]}{n}, \\ v(\ell \cdot (n,\nu)) &= -\frac{[(n,\nu)-1]}{n}, \qquad 0 < \ell < \frac{n}{(n,\nu)}, \\ v(j) &= 1/n \quad \text{otherwise.} \end{aligned} \tag{5.8}$$

The sum of any $(n,\nu)$ (circularly) consecutive values of $v$ excluding (respectively, including) $v(0)$ equals one (respectively, one); it follows that $Tv = \chi_{\{-r,\ldots,r\}}$. We show next that $Tu = v$, completing the proof that $TUT = T$. A key observation in this regard is that if $0 < \ell \leq n/(n,\nu)$, then

$$\begin{aligned} u(\ell\nu - (r+1)) &= u\left(-\left(\left[\frac{n}{(n,\nu)} + 1 - \ell\right]\nu - r\right)\right) = u\left(\left[\frac{n}{(n,\nu)} + 1 - \ell\right]\nu - r\right) \\ &= \frac{1}{n\nu} + \frac{(n,\nu)}{n}\left[\frac{n}{(n,\nu)} + 1 - \ell - \frac{1}{2}\right] - \frac{1}{2} \end{aligned}$$

holds in addition to (5.5), so that

$$u(\ell\nu - (r+1)) + u(\ell\nu - r) = \frac{2}{n\nu}$$

is constant in $\ell$.

We consider three cases.

(a) If $j \not\equiv 0 \pmod{(n,\nu)}$, then all $\nu$ terms in $(Tu)(j) = \sum_{k=-r}^{r} u(j+k)$ equal $1/(n\nu)$ except for $\nu/(n,\nu)$ pairs of the form $(u(\ell\nu-(r+1)),\ u(\ell\nu-r))$. Thus $(Tu)(j) = 1/n = v(j)$.

(b) If $j \equiv \ell\nu \pmod{n}$ with $0 < \ell < n/(n,\nu)$, then $(Tu)(j) = 1/n + u(\ell\nu - r) - u((\ell+1)\nu - r) = 1/n - (n,\nu)/n = v(j)$.

(c) If $j \equiv 0 \pmod{n}$, then $(Tu)(j) = 1/n + u(-r) - u(\nu - r)$ is larger than in case (b) by one, and so equals $v(0)$.

Thus $TUT = T$. Since $T$ and $U$ are circulants, they commute. There remains only the proof that $UTU = U$. For this we note first that

$$(I - TU)e_0 = e_0 - v = n^{-1}\,[(n,\nu)x - \mathbf{1}] \tag{5.9a}$$

where the vector $v$ is defined by (5.8) and the vector $x$ by

$$x(k) = \begin{cases} 1 & \text{if } k \equiv 0 \pmod{(n,\nu)}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.9b}$$

Now $(Ux)(j) = \sum_{k \equiv -j \pmod{(n,\nu)}} u(k)$ has the value $(n/(n,\nu)) \cdot (1/n\nu) = 1/\nu(n,\nu)$ if $j$ is congruent to neither $\pm r$ (mod $(n,\nu)$) and the (same!) value

$$\sum_{\ell=1}^{n/(n,\nu)} \left[\frac{1}{n\nu} + \frac{(n,\nu)}{n}\left(\ell - \frac{1}{2}\right) - \frac{1}{2}\right] = \frac{1}{\nu(n,\nu)}$$

otherwise. Thus the constant value of $U\mathbf{1}$ is $\sum_{k\in\mathbb{Z}_n} u(k) = (n,\nu) \cdot (1/\nu(n,\nu)) = 1/\nu$ and

$$(U - UTU)e_0 = n^{-1}\left[(n,\nu)\, Ux - U\mathbf{1}\right] = 0.$$

Since $U$ is circulant, it follows that $U = UTU$.

From (5.9) it follows that for general $g \in \mathbb{C}^n$

$$((I - TU)g)(j) = \frac{(n,\nu)}{n}\left[\sum_{k\equiv j \ (\mathrm{mod}(n,\nu))} g(k) - \frac{g(+)}{(n,\nu)}\right], \qquad j \in \mathbb{Z}_n, \tag{5.10}$$

and (2.47) above follows.

From the equality of (5.7) and (2.17) we obtain the complement

$$u(k) = \frac{1}{n}\left(\frac{1}{\nu} + \sum_{j=1}^{n-1}\left[\frac{\sin\left(\frac{\pi j \nu}{n}\right)}{\sin\left(\frac{\pi j}{n}\right)}\right]^{\div} \cos\left(\frac{2\pi j k}{n}\right)\right), \qquad k \in \mathbb{Z}_n,$$

to (4.14) for the present noninvertible transform.

**Acknowledgments.** I express my thanks to Professors Persi Diaconis, Peter Sarnak, Charles Stein, and Michael Wichura for helpful discussions related to this paper.

## REFERENCES

[1] T. W. ANDERSON, *The Statistical Analysis of Time Series,* John Wiley, New York, 1971.
[2] E. BATSCHELET, *Circular Statistics in Biology,* Academic Press, London, 1981.
[3] P. J. DAVIS, *Circulant Matrices,* John Wiley, New York, 1979.
[4] P. DIACONIS AND R. GRAHAM, *The Radon transform on $\mathbb{Z}_2^k$,* Pacific J. Math., 118 (1985), pp. 323–345.
[5] J. A. FILL, *Bounds on the coarseness of random sums,* Technical Report 208, University of Chicago, Chicago, IL, 1987.
[6] ———, *The Radon transform on the circle,* unpublished manuscript, 1987.
[7] ———, *Invertibility and nullities for Radon transforms on $\mathbb{Z}_n$ based on translates of sets of size 2 and 3,* unpublished manuscript, 1987.
[8] D. FREEDMAN AND D. LANE, *The empirical distribution of Fourier coefficients,* Ann. Statist., 8 (1980), pp. 1244–1251.
[9] S. HO AND L. H. Y. CHEN, *An $L_p$ bound for the remainder in a combinatorial central limit theorem,* Ann. Probab., 6 (1978), pp. 231–249.
[10] K. V. MARDIA, *Statistics of Directional Data,* Academic Press, London, 1972.
[11] A. WALD AND J. WOLFOWITZ, *Statistical tests based on permutations of the observations,* Ann. Math. Statist., 15 (1944), pp. 358–372.
[12] L. C. WASHINGTON, *Introduction to Cyclotomic Fields,* Springer-Verlag, New York, 1982.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 285-292, August 1989

# SORTING IN AVERAGE TIME $o(\log n)$*

M. AJTAI†, D. KARABEG‡, J. KOMLÓS§, AND E. SZEMERÉDI¶

**Abstract.** This paper presents a comparison sorting algorithm for the abstract CRCW PRAM parallel computer model. The algorithm sorts $n$ input values using $n$ processors and runs in time $O(\log n/\log \log n)$ on the average, assuming that all permutations of the input are equally likely.

**Key words.** sorting, average-time complexity, parallel algorithms

## 1. Introduction.

The parallel computer model we will be using is a powerful variant of the concurrent-read-concurrent-write parallel random access machine called the Abstract PRAM. It consists of an unbounded number of random-access memory cells, each of which can store an unbounded number of bits, and of a set of $N$ processors $\{P_1, \cdots, P_N\}$. In one unit of time each processor can read one arbitrary memory cell (possibly concurrently with other processors); do an arbitrary computation on the contents of its unbounded internal memory; and write to an arbitrary memory location. If several processors write to the same location, the one with the largest processor number will succeed.

When a computation is initiated, all the processors start executing their programs concurrently and synchronously. The computation terminates when all the processors terminate.

The unbounded computing power of the processors enables us to de-emphasize the cost of computation entirely, and to concentrate on the communication issues.

We say that a parallel algorithm solves the sorting problem if, when given $n$ values from an arbitrary ordered set in the first $n$ memory cells $M[1, \cdots, n]$ as input, on termination of the algorithm the first $n$ memory cells contain the input values in natural order.

It can be immediately observed that any function of the input can be computed in $O(\log n)$ time ($n$ is the size of the input measured as the number of memory cells that contain the input), if the number of processors is $O(n)$. Indeed, it suffices to merge the input values into pairs iteratively, until all the inputs are contained in one memory cell, and then read them into one processor that can then compute an arbitrary function in unit time. Consequently, we will be interested in studying the possibility of sublogarithmic-time computation.

It is also easy to see that an increased processor power is necessary if we want to sort in sublogarithmic time with $O(n)$ processors. The $\Omega(n \log n)$ comparisons required for sorting give an immediate $\Omega(\log n)$ lower bound for any parallel machine with $n$ processors whose processing power is limited to doing one comparison at a time. On the other hand, sorting in logarithmic time can be done even with a much less powerful, "realistic," computer model (the parallel sorting network of Ajtai, Komlós, and Szemerédi [AKS]). Concurrent reads and writes are also necessary—the $\Omega(\log n)$ lower bound for the exclusive write PRAM model for computing the logical OR function (see Cook,

* Received by the editors January 4, 1988; accepted for publication (in revised form) February 1, 1989.

† IBM, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099.

‡ Computer Science Department, University of Southern California, University Park, Los Angeles, California 90089-0782, and Rudjer Bošković Institute, Zagreb, Yugoslavia.

§ Department of Mathematics, Rutgers University, New Brunswick, New Jersey 08903.

¶ Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903, and Hungarian Academy of Sciences, Budapest, Hungary.

Dwork, and Reischuk [CDR]) yields the same lower bound for sorting. On the CRCW PRAM the logical OR can be computed in constant time in an obvious way.

A lower bound $\Omega(\sqrt{\log n})$ for sorting on CRCW PRAM with unbounded processor power has been proved by Meyer auf der Heide and Wigderson [MW]. They have conjectured that the tight lower bound is $\Omega(\log n)$. Beame and Hastad [BH] have recently proved the lower bound $\Omega(\log n/\log\log n)$ for parity on the same machine model, which implies the same lower bound for sorting. Karabeg [K] has pointed out that the same lower bound holds for computing parity (and sorting) with probability $\frac{1}{2} + \varepsilon$.

On the other hand, even the prefix summation problem recently has been shown to have time complexity $O(\log n/\log\log n)$ with a linear number of processors (Cole and Vishkin [CV]; see also the earlier paper by Reif [R]).

We remark that using more than $n$ processors it is easy to sort in sublogarithmic time. We can use the AKS sorting network and take advantage of the extra processors to "look into the future" by comparing an input to multiple inputs in parallel. (Teams of processors are used for each multiple comparison. Each team is assigned the task of verifying in constant time one of the possible outcomes of the multiple comparison.) This results in a $O(\log n/\log\log k)$-time sorting algorithm if the number of processors is $kn$.

In this paper we present a sorting algorithm using $n$ processors that runs in $O(\log n/\log\log n)$ time for most inputs (and also on the average, assuming that all input order types are equally probable).

**2. The algorithm.** With $n^2$ processors, sorting is as easy as counting because we can assign $n$ processors to every input element that compares the element with all others. Counting, done by the mentioned prefix summation algorithm, provides the rank. We will call such a procedure QSORT.

The idea behind our sorting algorithm is to select a large subset of the inputs, sort them using QSORT, and compare the rest of the elements to them in parallel. (This, in a way, resembles Quicksort.) To do these latter comparisons fast enough, a $k$-ary search tree is built on the sorted elements, where $k$ is equal to $\log n$. When searching through the $k$-ary tree, a processor can take advantage of its unbounded computing power to compare an input element to $k$ keys at a time, thus gaining a factor $\log k$ in comparison to a binary tree search. We shall call the selected and sorted inputs *bucket separators*, or *separators* for short, and the rest of the inputs will be called *bucket elements*. All bucket elements whose values range between two adjacent bucket separators form a common bucket.

The above procedure can now be applied to each bucket iteratively until all the buckets are of a "manageable" size with a large probability. At that point the elements within each bucket are sorted and counted. The prefix sums of the bucket sizes are produced next. The rank of each element is computed by adding its rank within its bucket to the prefix sum that corresponds to the sum of the sizes of buckets with smaller elements.

**2.1. Procedure SIFT.** The partition of input elements into buckets and bucket separators is produced and refined by iterative application of procedure SIFT. It consists of four phases. The first phase selects a subset of each bucket, of size roughly equal to the square root of the bucket size. This selection is done as follows. In every round of SIFT we define a hash function by reading in new bits of the addresses of the input elements. Then every element attempts to write into the location corresponding to the value of its hash function in a separate table reserved for its bucket. The one that succeeds is selected.

The selected elements become new bucket separators. In the second phase the new bucket separators are sorted by using the QSORT procedure. In the third phase the sorted separators are composed into a $k$-ary search tree. In the fourth phase each bucket element is compared to the search tree and new buckets are formed of elements that belong to the same leaf of the search tree.

The shared memory is divided into three segments, denoted $I$, $S$, and $W$. The segments are of size $n$, and they represent the space for the input elements, the space for the bucket separators, and the "scratch-pad" working space, respectively.

A location in the $S$ segment is assigned to each bucket separator so that a larger address corresponds to a larger separator. Each bucket is represented by a *bucket identifier* $l$ that is the address within the $S$ segment (an integer from 0 to $n - 1$) of the location assigned to the smaller of the two separators that define the bucket. The bucket identifier is known to each element of the bucket. Before the first iteration of the algorithm SIFT is started, the whole input is one bucket, with identifier 0.

All processors execute each iteration of SIFT in synchrony, all working on the same phase at the same time, except some of them possibly being idle.

ALGORITHM SIFT($i$).

```
Phase 1. $P_j$ reads the next $d$ digits of $j$, where $d = \lfloor 2^{-i} \log n \rfloor$. These
$d$ digits describe a number $r$ in binary.

$P_j$ attempts to write $x_j$ into cell $W[l(x_i) + r]$. Each processor reads the
same location to verify if the write was successful, and if yes,
labels $x_j$ as a bucket separator.

Phase 2. QSORT bucket separators. Assign bucket identifiers to
bucket separators.

Phase 3. Within each bucket, build a k-ary tree from the new
bucket separators in $W$-space.

Phase 4. Processors search through the k-ary tree in parallel to
find their refined bucket id.
```

**2.2. Procedure COLLECT.** After the division of elements into buckets has been sufficiently refined by successive application of procedure SIFT, the elements that belong to a common bucket need to be sorted. Even though the buckets are now small and could be sorted with an $O(\log n)$ sorting algorithm, the problem of sorting the buckets is still not trivial because it is not *globally* known which inputs belong to the same bucket. Consequently it is not obvious how to organize a collection of processors to execute an algorithm on the elements that belong to a common bucket. This gives rise to the following problem.

DEFINITION 2.1. We say that an algorithm solves the *collection* problem if, when presented with $n$ values in $M[1, \cdots, n]$ on input, each input value consisting of a data field from an ordered set and a key in $[1, \cdots, n]$, it produces some encoding of each set of inputs with the same key $i$ (buckets) in memory cell $M[i]$.

We propose the following algorithm that has a very good expected behavior. The algorithm will build a binary search tree on the elements with the same key and then proceed through the tree bottom-up merging the child vertices with their parent.

Input in $M[i]$ is associated with processor $P_i$. Note that by having a processor read the cell it is about to write, new information can be written into memory cells without destroying the existing information.

ALGORITHM COLLECT.

Phase 1. All processors in parallel: processors whose associated inputs have a key equal to $i$ all attempt to write their processor number and data value into the same memory location $M[i]$. This selects one processor with index $j = j(i)$ for every $i$.

All other processors with key $i$ change their key to $j$, and compare their data value to that in $M[i]$. The ones with a smaller value attempt to write into memory $j$, this way selecting the left child of $j$. The right child is selected in the same way.

Repeat the above recursively.

Phase 2. After the trees are constructed for all buckets, traverse the trees bottom-up and merge the data at each step.

**2.3. Algorithm PSORT.** We are now ready to describe the PSORT algorithm for sorting $n$ inputs with $n$ processors. The algorithm consists of four phases. In the first phase the partition of the inputs into bucket elements and bucket separators is produced and refined by successive application of procedure SIFT. Next, the bucket elements are brought together by procedure COLLECT. In the end, the prefix sums of the sizes of buckets are computed, and the rank of each input is set to be equal to the sum of its intrabucket rank and the appropriate prefix sum. Finally, each input element $x_i$ is repositioned to a location whose address is equal to the rank of $x_i$.

ALGORITHM PSORT.

Phase 1. Perform SIFT$(i)$ iteratively for $i = 1, \cdots, \log \log n$.

Phase 2. Perform COLLECT on the input elements, with the input values as the data items, and the bucket identifiers provided by SIFT as keys.

Phase 3. Produce the sizes of the buckets in the corresponding locations in $S$ segment. Perform on the $S$ segment an algorithm that produces the prefix sums.

Phase 4. Each processor $P_i$ reads the prefix sum from the location $S[l(x_i)]$ and learns the rank of $x_i$ within its own bucket from the result of COLLECT. $P_i$ produces the final rank$(x_i)$.

Phase 5. Each processor $P_i$ stores input value $x_i$ into address rank$(x_i)$.

**3. Analysis of PSORT.** In this section we give a sequence of results that lead to the conclusion that, with an overwhelming probability, the algorithm PSORT terminates in $O(\log n/\log \log n)$ time. We first show that the iterations of SIFT in Phase 1 of PSORT terminate in $O(\log n/\log \log n)$ time. Subsequently we show that, with a large probability, the largest bucket size $s$ at the start of the COLLECT procedure is small compared to $n$, or, more precisely, that $\log \log s = o(\log \log n)$. This will allow us to conclude that COLLECT terminates in $O(\log n/\log \log n)$ time with a large probability. The final stage, namely computation of the prefix sums and the ranks of the inputs, is done in $O(\log n/\log \log n)$ time by using a known parallel summation algorithm [CV].

**3.1. Analysis of SIFT.** As explained earlier, the algorithm SIFT is applied iteratively to produce a sufficiently fine partition of the input elements into buckets. Here we first show that the time required for the $\log\log n$ iterations of SIFT as performed in Phase 1 of PSORT is $O(\log n/\log\log n)$. Subsequently, we show that the largest of the buckets produced as a result of Phase 1 of PSORT is small with a large probability.

LEMMA 3.1. *Phase* 1 *of* PSORT *terminates in* $O(\log n/\log\log n)$ *time.*

*Proof.* Phase 1 of SIFT($i$) requires only a constant time, Phase 3 terminates in $O(\log\log n)$ time, and Phases 2 and 4 each require no more than $O(\log w/\log\log w)$, where $w$ is the size of the region in the working space used for selecting the bucket separators, $w = n^{1/2^i}$. Hence, the $i$th iteration of SIFT requires no more than $O((1/2^i)(\log n/\log\log n - i) + \log\log n)$ time. The result follows by summing the values of this expression for $i$ ranging from one to $\log\log n$. □

To derive a bound on the size of the largest bucket on termination of Phase 1 of PSORT, we need the following elementary technical result.

LEMMA 3.2. *Given a set $S$ of size $s$ and a set of $l$ labels such that $s \geqq l$, a subset of $S$ is selected as follows. A label for each element of $S$ is selected at random with replacement, and from each subset of elements with the same label one element is selected at random. Then the probability that any given element of $S$ is selected exceeds $l/3s$.*

*Proof.* Note that the number of selected elements will be equal to the number of selected labels. If $i$ labels are selected, the probability that an arbitrary element $x \in S$ is selected is equal to $i/s$ since, by symmetry, all elements have the same probability of being selected. Let $\delta$ be a real constant, $0 \leqq \delta \leqq 1$. Then

$$P(x \text{ is selected}) = \sum_{i=1}^{l} \frac{i}{s} P(i \text{ labels are selected})$$

$$\geqq \frac{\delta l}{s} \sum_{i=\delta l}^{l} P(i \text{ labels are selected}).$$

This probability can be estimated as follows:

$$P(x \text{ is selected}) \geqq \max_{\delta} \frac{\delta l}{s} P(\delta l \text{ or more labels are selected})$$

$$= \max_{\delta} \frac{\delta l}{s} [1 - P(\text{less than } \delta l \text{ labels are selected})]$$

$$= \max_{\delta} \frac{\delta l}{s} \left(1 - \binom{l}{\delta l - 1} \left(\frac{\delta l - 1}{l}\right)^{s}\right)$$

$$> \max_{\delta} \frac{\delta l}{s} \left(1 - \binom{l}{l/2} \delta^{l}\right).$$

By choosing $\delta = \frac{1}{2}$ and applying the Stirling's formula, we have

$$P(x \text{ is selected}) > \frac{l}{2s}\left(1 - \frac{1}{2\pi}\sqrt{\frac{4}{l}}\right) > \frac{l}{3s}$$

as claimed. □

We now turn to the bound on the largest bucket size.

LEMMA 3.3. *Let $B$ denote the size of the largest bucket at the end of Phase* 1 *of* PSORT, *and let $n$ be the number of input elements. Then the probability that $B$ exceeds $2(3c \log n)^{\log\log n}$ is smaller than $n^{2-c}$.*

*Proof.* Note that, by symmetry, in any iteration of SIFT all the elements within a bucket have the same probability of being selected as separators. Also note that the probability that a given element is selected is smaller if the size of its bucket is larger. Consequently, the elements of the largest bucket will be the least likely to be selected as separators.

Let $\hat{P}_i$ be the minimum over all input elements $x$ of the probability that $x$ is selected as a bucket separator during the first $i$ iterations of SIFT. Let $B_i$ be the size of the largest bucket after the $i$th iteration of SIFT. Then by the above argument and Lemma 3.2,

$$\hat{P}_i \geqq \hat{P}_{i-1} + \frac{n^{1/2^i}}{3B_{i-1}}. \tag{1}$$

We now claim that

$$B_i \leqq \frac{c}{\hat{P}_i} \log n \tag{2}$$

with an overwhelming probability that depends on the constant $c$. This bound on $B_i$ implies the desired bound on the final bucket size $B$. By (1) and (2)

$$B_i \leqq 3cB_{i-1} n^{-1/2^i} \log n, \tag{3}$$

$$B_0 = n, \tag{4}$$

hence, for $i = \log\log n$,

$$B_i = B \leqq (3c \log n)^{\log\log n} n^{1 - \sum_{j=1}^{\log\log n} 2^{-i}} \tag{5}$$

and the required bound

$$B \leqq 2(3c \log n)^{\log\log n} \tag{6}$$

easily follows.

To estimate the bound on the probability that (2) is violated, let us consider the probability $Q_i(j)$ that some given $j$ consecutive elements (in the natural order) are not selected for separators in the first $i$ rounds. Clearly,

$$Q_i(j) \leqq (1 - \hat{P}_i)^j. \tag{7}$$

Substituting the expression for $B_i$ from (2) for $j$, we obtain

$$Q_i\left(\frac{c \log n}{\hat{P}_i}\right) \leqq (1 - \hat{P}_i)^{c \log n / \hat{P}_i} \leqq e^{-c \log n} = n^{-c}. \tag{8}$$

Since there are no more than $n$ sequences of consecutive input elements of the specified length, the probability that (2) is violated for *some* sequence is at most $n^{1-c}$. Consequently, the probability that (2) is violated in some of the log log $n$ iterations of SIFT is at most $n^{(1-c)} \log\log n$, which is smaller than $n^{2-c}$. □

**3.2. Analysis of COLLECT.** The running time of COLLECT depends on the size of its input buckets. Consequently, a bound on the running time of COLLECT will be shown to hold with an overwhelming probability under the assumption that the bucket sizes are bounded.

LEMMA 3.4. *Let $n$ input elements be distributed into buckets so that the size $B$ of the largest bucket satisfies the inequality*

$$\log\log B \leqq \tfrac{1}{2} \log\log n. \tag{9}$$

*Then the probability that the time $T(n)$ required for running the algorithm* COLLECT *on all the buckets in parallel exceeds $c(\log n/\log\log n)$ is smaller than $c_1 n^{1-c/3}$, where $c_1$ is a constant.*

*Proof.* Observe that the selection in Step 1 of COLLECT of the bucket element subsequently used to subdivide the bucket is such that all the elements have an equal chance of being selected (we assume that all input permutations are equally likely). We can model this random selection by a uniformly distributed random variable $u$, $u \in (0, 1]$. For a bucket of size $B$, the bucket element that corresponds to a given value of $u$ is determined by its rank $i$ within the bucket, $i = \lceil uB \rceil$.

Algorithm COLLECT consists of two phases, one in which the bucket elements are organized into a binary search tree, and one in which the tree is traversed bottom-up. A constant number of steps is spent on each level of the longest directed path of the tree in each phase. Thus, the time the algorithm requires to collect one bucket is a constant multiple of the height of the tree that corresponds to that bucket. Since all the buckets are collected in parallel, the total running time of COLLECT is a constant multiple of the height of the highest search tree corresponding to a bucket. Hence, it suffices to exhibit a bound on the maximum height of a search tree.

Let $P_x(d)$ be the probability that a specific leaf $x$ is at the level $d$ or higher. Let $u_i$ be the random variable corresponding to the selection of the internal node (the subdividing element) of the search tree at level $i$ of the directed path of the tree corresponding to leaf $x$. Then

$$\begin{aligned} P_x(d) &\leqq P\left(\prod_{i=1}^{d} u_i > \frac{1}{B}\right) \\ &= P\left(e^{\sum_{i=1}^{d} \log u_i} > e^{-\log B}\right) \\ &= P\left(e^{t\sum_{i=1}^{d} \log u_i} > e^{-t\log B}\right) \end{aligned}$$

for any real constant $t > 0$. Since by Markov's inequality $P(x > c) \leqq Ex/c$ (cf. [Rn, p. 85]), then

$$(10)\qquad \begin{aligned} P_x(d) &\leqq \frac{Ee^{t\sum_{i=1}^{d} \log u_i}}{e^{-t\log B}} \\ &= B^t E \prod_{i=1}^{d} e^{t \log u_i} = B^t (Eu^t)^d = \frac{B^t}{(1+t)^d}. \end{aligned}$$

Here we used the fact that all random variables $u_i$ are independent and uniformly distributed.

By computing the minimum of the right-hand side of (10) with respect to $t$ we find that the best bound is obtained for

$$(11)\qquad t = \frac{d - \log B}{\log B}.$$

Substituting (11) into (10), we have

$$(12)\qquad P_x(d) \leqq \exp\left(-\left(d \log\left(\frac{d}{\log B}\right) - d + \log B\right)\right).$$

Since, by assumption, $\log\log B \leqq \frac{1}{2}(\log\log n)$, then

$$(13)\qquad P_x\left(c\frac{\log n}{\log\log n}\right) < e^{-(1/3)c\log n} = n^{-c/3}.$$

There are less than $n$ leaves in all trees. Hence, the probability that some leaf is on level $(c(\log n/\log\log n))$ or higher does not exceed

$$nP_x\left(c\frac{\log n}{\log\log n}\right), \tag{14}$$

which is smaller than $n^{-(c/3-1)}$. □

The results of this and the previous section allow us to conclude the following theorem.

THEOREM 3.1. *The probability that the running time of* PSORT *on an input of size $n$ exceeds $C(\log n/\log\log n)$ is less than $n^{-C'}$ where $C'$ is proportional to $C$.*

**4. Concluding remarks.** We have described a deterministic sorting algorithm called PSORT that sorts most of the inputs in time $o(\log n)$. It is easy to see how PSORT can be modified into a probabilistic sorting algorithm that sorts *any* input in expected time $o(\log n)$.

By reviewing the existing upper bound results in parallel computing it can be observed that, with few exceptions, the probabilistic upper bounds match the corresponding deterministic upper bounds. Techniques that have been developed in recent years for "simulating randomness" usually allow for constructing efficient deterministic algorithms whenever there is an efficient probabilistic algorithm for the same problem. Whether or not such techniques can be effective in the present case remains an open question. If the $\Omega(\log n)$ lower bound conjectured in [MW] turns out to be exact, this problem will be one of the few in parallel computing for which the probabilistic upper bound is smaller than the deterministic lower bound.

**Acknowledgment.** We are indebted to Jeff Kahn for several fruitful discussions on the topic.

## REFERENCES

[AKS] M. AJTAI, J. KOMLÓS, AND E. SZEMERÉDI, *Sorting in c* log *n parallel steps*, Combinatorica, 3 (1983), pp. 1–19.

[BH] P. BEAME AND J. HASTAD, *Optimal bounds for decision problems on the* CRCW PRAM, in Proc. 19th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, NY, 1987, pp. 83–93.

[CDR] S. COOK, C. DWORK, AND R. REISCHUK, *Upper and lower time bounds for parallel random access machines without simultaneous writes*, J. Algorithms, to appear.

[CV] R. COLE AND U. VISHKIN, *Approximate and exact parallel scheduling with applications to list, tree and graph problems*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 1986, pp. 478–491.

[K] D. KARABEG, *Sublogarithmic-time parallel algorithms*, Ph.D. dissertation, University of California, San Diego, CA, June 1988.

[MW] F. MEYER AUF DER HEIDE AND A. WIGDERSON, *The complexity of parallel sorting*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 1985, pp. 532–540.

[R] J. REIF, *An optimal parallel algorithm for integer sorting*, in Proc. 26th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 1985, pp. 496–503.

[Rn] A. RÉNYI, *Foundations of Probability*, Holden-Day, San Francisco, CA, 1970.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 293-299, August 1989

# LEAF PAIRS AND TREE DISSECTIONS*

STEPHEN F. ALTSCHUL†

**Abstract.** A question from biological sequence comparison can be formulated as a simple problem in matrix algebra. While this problem can be solved by matrix inversion, its special structure permits a computationally more efficient solution by means of graph theory.

**Key words.** graph theory, trees, matrix algebra, sequence comparison

**AMS(MOS) subject classifications.** 05C05, 15A24, 68E10, 92A90

**Introduction.** An important question in biological sequence comparison is how to align two or more sequences so as to elucidate their similarities. Dynamic programming algorithms are generally recognized as the method of choice for aligning two sequences (Needleman and Wunsch [7], Sellers [10], Waterman [11]). The generalization of this approach to multiple sequences that is closest to biological intuition computes a multiple alignment that minimizes the cost of an evolutionary tree (Sankoff [8], Sankoff and Cedergren [9], Altschul and Lipman [3]). A variation that permits algorithms requiring much less space and time computes a multiple alignment that minimizes the sum of the costs of all imposed pairwise alignments (Carrillo and Lipman [4], Altschul and Lipman [3], Altschul [1]). For this strategy to yield biologically reasonable results, the costs of the various pairwise alignments need to be given different weights (Altschul, Carroll, and Lipman [2]). The reason is best illustrated by considering an alignment of three sequences $A$, $B$, and $C$. Imagine including several sequences very similar to $A$ in the multiple alignment. If all pairwise alignments are given equal weight, then the many pairs similar to $A - B$ and $A - C$ will outvote the single $B - C$ pair. Sequence $A$ will essentially dictate the multiple alignment simply because there are several copies of it in the data. Since most any set of related DNA or protein sequences will contain some sequences more closely related to one another than to the rest, this problem remains even if extra copies of virtually identical sequences are removed. A way is needed to weight the pairwise alignments so that redundant information is discounted.

Pairwise distance information can be used to construct an unrooted evolutionary tree relating a set of sequences so that the leaves of the tree correspond to the input sequences (Felsenstein [6]). Each edge of the tree is assigned a length that estimates the amount of change occurring along that edge. Altschul, Carroll, and Lipman [2] argue that, given such a tree, it is possible to define correlation coefficients for the pairwise alignments. Finding appropriate pair weights for use in constructing a multiple alignment then reduces to a problem in matrix algebra which can be solved by simple matrix inversion. However, the given evolutionary tree imposes a special structure on the matrix in question. This permits a graph theoretical solution which requires time that is only linear in the dimension of the matrix. Certain interesting graph theoretical constructs arise from the consideration.

Our problem is analogous to one considered by Felsenstein [5], in which he seeks weights for the individual leaves of a rooted evolutionary tree. His formulation also gives rise to a problem that can be solved by matrix inversion but whose special structure

* Received by the editors July 18, 1988; accepted for publication (in revised form) December 6, 1988.

† Mathematical Research Branch, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health. Present address, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Building 38A, Room 8N-811, Bethesda, Maryland 20894.

permits a linear time solution. The parallel between his results and ours suggests the possibility of a more general theorem.

**The problem.** In all that follows, we will assume that we have been given a tree $T$ with edge set $E$, and vertices divided into a set of leaves $L$ and a set of internal vertices $N$. We define $P$ as the set of unordered pairs of distinct leaves. To each $p \in P$ there corresponds a unique path through the tree. Call the set of internal vertices in this path $N_p$, the set of edges in the path $E_p$, and denote by $E_r$ the set of edges in an arbitrary path $r$.

Assume each edge $e \in E$ has a length $l_e$. Extend the concept of length by letting $l_p = \sum_{e \in E_p} l_e$ and $l_{pq} = \sum_{e \in E_p \cap E_q} l_e$ for all pairs $p, q \in P$.

Define the symmetric matrix $\mathbf{M}$ with rows and columns indexed by the set $P$ by $\mathbf{M}_{pq} = (l_{pq}^2 / l_p l_q)$. We seek a solution to the matrix equation

$$\mathbf{M}\vec{x} = \vec{u} \tag{1}$$

where $\vec{u}$ is a vector of 1's.

It is possible to solve for $\vec{x}$ simply by writing $\vec{x} = \mathbf{M}^{-1}\vec{u}$. To calculate $\vec{x}$ this way, however, requires inverting a $|P| \times |P|$ matrix. Just to write down the matrix requires $O(|P|^2)$ time. We show below that the special structure of this problem allows us to solve for $\vec{x}$ in $O(|P|)$ time.

**A graph theoretical solution.**

DEFINITION 1. An *edge choice* $c: N \rightarrow E$ is an injection such that for all $v \in N$, $c(v)$ is adjacent to $v$. For trees with no internal vertices (i.e., those having one or two vertices), the function with vacuous domain and range is an edge choice. Let $C$ be the set of all edge choices.

DEFINITION 2. A *dissection* $D \subset E$ is the complement of the range of an edge choice. $S$ is the set of all dissections of the tree $T$.

Intuitively, a dissection is the set of edges that remain after each internal vertex eliminates an adjacent edge, with the constraint that no two vertices eliminate the same edge.

LEMMA 1. *For all dissections $D \in S$ and pairs $p \in P$, $D \cap E_p$ is nonempty.*

*Proof.* Edges of $E_p$ can be eliminated only by the $|E_p| - 1$ internal vertices of path $p$.

LEMMA 2. *For all dissections $D \in S$ and all internal vertices $v \in N$, there is a path $r$ from $v$ to a leaf such that $D \cap E_r$ is empty.*

*Proof.* Let $v \in N$, $D \in S$, and let $c$ be an edge choice giving rise to $D$. Let $v_1$ be the vertex other than $v$ that is adjacent to edge $c(v)$. If $v_1 \in N$, let $v_2$ be the vertex other than $v_1$ that is adjacent to $c(v_1)$. Repeat this process until $v_n$ is a leaf. None of the edges of the path from $v$ to $v_n$ are members of $D$.

LEMMA 3. *Each dissection $D$ arises from a unique edge choice $c_D$.*

*Proof.* Suppose there are two distinct edge choices giving rise to dissection $D$. Let $v \in N$ be a vertex on which they differ. Using the construction in the proof of Lemma 2, the two edge choices give rise to disjoint paths satisfying Lemma 2. The union of these two paths violates Lemma 1.

DEFINITION 3. For a pair $p \in P$, a *p-edge choice* is an edge choice that maps $N_p$ into $E_p$. A *p-dissection* is the complement of the range of a $p$-edge choice. $C_p \subset C$ is the set of all $p$-edge choices and $S_p \subset S$ is the corresponding set of all $p$-dissections.

Any $p$-edge choice $c$ can be decomposed into two injections $c_1: N_p \rightarrow E_p$ and $c_2: N - N_p \rightarrow E - E_p$ each of which maps internal vertices to adjacent edges. Conversely, any $c_1$ and $c_2$ of this description can be paired to yield a $p$-edge choice $c$. Observe that since $|N_p| = |E_p| - 1$, the complement of the range of any $c_1$ is a single edge of $E_p$.

Furthermore, each edge of $E_p$ clearly corresponds to a unique $c_1$. Finally, since the range of $c_2$ is $E - E_p$, we must have $|D \cap E_p| = 1$ for all $p$-dissections $D$.

DEFINITION 4. For a pair $p \in P$, a *p-elimination* is the set $D - D \cap E_p$ for some $p$-dissection $D$. $S'_p$ is the set of all $p$-eliminations.

LEMMA 4. *If $D'$ is a p-elimination, then for any edge $e \in E_p$, $D' \cup \{e\}$ is a p-dissection. Furthermore, these are the only p-dissections giving rise to $D'$.*

*Proof.* Notice that a $p$-elimination $D'$ is just the complement in $E$ of the range of some $c_2$. Pairing this $c_2$ with any $c_1$ yields an edge choice whose $p$-dissection gives rise to $D'$. These $p$-dissections are just $D'$ adjoined with the various edges of $E_p$.

DEFINITION 5. The *weight* $w_T$ of tree $T$ is given by the formula $w_T = \sum_{D \in S} \prod_{e \in D} l_e$. For $T$ consisting of a single vertex, $w_T = 1$.

DEFINITION 6. The *weight* $w_p$ of a pair $p \in P$ is given by the formula $w_p = \sum_{D \in S_p} \prod_{e \in D} l_e$.

DEFINITION 7. The *bias* $b_p$ of a pair $p \in P$ is given by the formula $b_p = \sum_{D' \in S'_p} \prod_{e \in D'} l_e$.

LEMMA 5. *For all $p \in P$, $w_p = b_p l_p$.*

*Proof.* By Lemma 4, each $p$-elimination $D'$ arises from $|E_p|$ $p$-dissections. Each of these $p$-dissections may be regenerated by adding a different member of $E_p$ to $D'$. The lemma follows immediately.

THEOREM 1. *For all $p \in P$, $\sum_{q \in P} l_{pq}^2 b_q = l_p w_T$.*

*Proof.* We shall show that each term (summand) on the left-hand side of the equation corresponds to a unique term on the right-hand side, and the converse. First, consider a left-hand term $t$. It has the form of a triple $l_{e_1} l_{e_2} \prod_{e \in D'} l_e$ where $e_1, e_2 \in E_p \cap E_q$ and $D'$ is a $q$-elimination. Since $e_2 \in E_q$, by Lemma 4 $e_2$ can be adjoined to $D'$ to yield a dissection. The product $l_{e_2} \prod_{e \in D'} l_e$ can therefore be rewritten as $\prod_{e \in D} l_e$ for some dissection $D$. In this way, the triple $t$ can be written as a unique product of a term from the formal sum $l_p$ and a term from $w_T$. Each left-hand term can thus be mapped to a unique right-hand term. It remains to show that every right-hand term can arise in this way from a left-hand term.

Consider any right-hand term $t$. It can be written in the form $l_{e_1} \prod_{e \in D} l_e$ for some edge $e_1 \in E_p$ and some dissection $D$. Along the path corresponding to $p$ name the vertices in order $v_1, \cdots, v_n$. Let $v_i$ and $v_{i+1}$ be the vertices at either end of $e_1$. By Lemma 3, let $c_D$ be the edge choice corresponding to $D$. Since $c_D$ is an injection, either $c_D(v_i) \neq e_1$ or $c_D(v_{i+1}) \neq e_1$. (We deem the condition satisfied by $v_i$ or $v_{i+1}$ a leaf, on which $c_D$ is not defined.) Without loss of generality, suppose $c_D(v_i) \neq e_1$. By Lemma 2, there is a path from $v_i$ to leaf $v_\alpha$ whose intersection with $D$ is empty. By Lemma 1, there exists a smallest integer $j > i$ for which $c_D(v_j)$ does *not* connect $v_j$ and $v_{j-1}$; if such an integer did not exist, the path from $v_n$ to $v_\alpha$ would have empty intersection with $D$. Denote by $e_2$ the edge between $v_j$ and $v_{j-1}$ (Fig. 1). By Lemma 2, there are paths $r_1$ and $r_2$ from either end of $e_2$ to a leaf, each of which has empty intersection with $D$. Since neither path uses edge $e_2$, they must be disjoint. Let $q$ be the pair of leaves these paths reach. Then $E_q = E_{r_1} \cup E_{r_2} \cup \{e_2\}$ and $D \cap E_q = \{e_2\}$. Therefore $D' = D - \{e_2\}$ is a $q$-elimination. Note that both $e_1$ and $e_2$ are members of $E_q$ and of $E_p$. The term $t$ can thus be written $l_{e_1} l_{e_2} \prod_{e \in D'} l_e$, where $D'$ is a $q$-elimination and $e_1, e_2 \in E_p \cap E_q$. This is a left-hand term.

THEOREM 2. *The vector given by $\vec{x}_p = w_p / w_T$ solves $\mathbf{M}\vec{x} = \vec{u}$.*

*Proof.* Dividing both sides of the equation of Theorem 1 by $l_p w_T$ and using Lemma 5 to replace $b_q$ by $w_q / l_q$ we get $\sum_{q \in P} (l_{pq}^2 / l_p l_q)(w_q / w_T) = 1$ for all $p \in P$.

COROLLARY 1. *If the lengths $l_e$ are all positive, the solution to $\mathbf{M}\vec{x} = \vec{u}$ has all components positive.*

*Proof.* All the summands in $w_p$ and $w_T$ are positive.

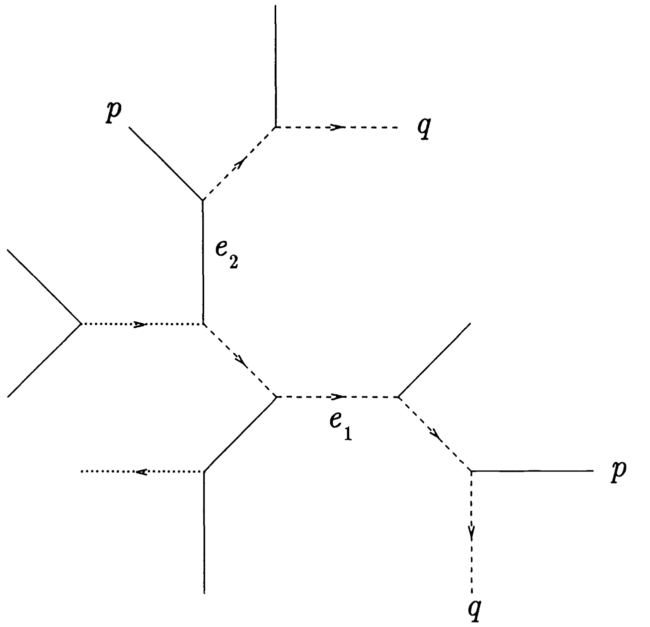


FIG. 1

**Time complexity.** Let us consider how much time is required to compute the $\vec{x}_p$ using the formula of Theorem 2. First, we prove a lemma that allows certain trees' weights to be computed recursively.

LEMMA 6. *Let $T$ be a tree and let $v \in N$. Suppose that all internal vertices of $T$ other than $v$ have degree at least three* (*i.e., they are incident to at least three edges*). *Let $T_1, \cdots, T_n$ be the subtrees of $T$ that hang from $v$ by the respective edges $e_1, \cdots, e_n$ and let $T'_i$ be the subtree $T_i$ minus the vertex $v$ and the edge $e_i$. Then $w_T = \sum_{i=1}^n (w_{T'_i} \prod_{j \neq i} w_{T_j})$. If all the $w_{T_i}$ are nonzero, this can be rewritten $w_T = (\sum_{i=1}^n w_{T'_i}/w_{T_i})(\prod_{i=1}^n w_{T_i})$.*

*Proof.* Let $S_i = \{D \in S | c_D(v) = e_i\}$. Clearly $S = \bigcup_{i=1}^n S_i$. While $v$ is an internal vertex in $T$, it becomes a leaf in the trees $T_1, \cdots, T_{i-1}, T_{i+1}, \cdots, T_n$. Let $v'$ be the vertex at the other end of edge $e_i$. The assumption that all internal vertices of $T$ other than $v$ have degree at least three implies that if $v'$ is an internal vertex in $T$ it remains one in $T'_i$. Therefore, any dissection $D \in S_i$ can be seen as the union of dissections of the trees $T_1, \cdots, T_{i-1}, T'_i, T_{i+1}, \cdots, T_n$ (Fig. 2). The lemma follows immediately from the definition of $w_T$.

In what follows we assume that the weights of the subtrees hanging from all vertices are nonzero. (By Corollary 1, this is true if all the $l_e$ are positive.) While the results hold in any case, this assumption allows us to use the second equation of Lemma 6, which simplifies several arguments.

LEMMA 7. *Given a tree $T$, all of whose internal vertices have degree at least three, the weights of all subtrees hanging from all vertices of $T$ can be computed in $O(|L|)$ time.*

*Proof.* For each edge $e \in E$ we need to compute two numbers: the weight of the subtree hanging by that edge from either of the two vertices it touches. Consider the following algorithm for computing these numbers $W_i$ and $V_i$:

(1) Choose an internal vertex $v_0$ of $T$. Starting with $v_0$, perform a depth first search for edges and vertices. Label them $e_1, \cdots, e_m$ and $v_1, \cdots, v_m$ in the order they are

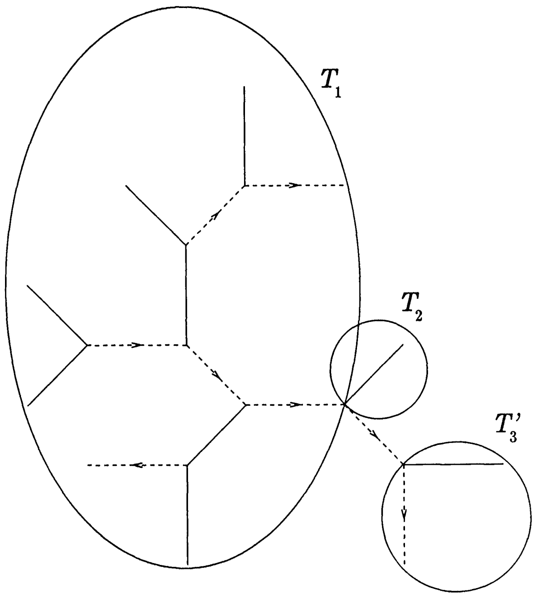


FIG. 2

found. For every edge $e_i$ found while searching $v_j$, set $f(i) := j$, and add $i$ to the list $\Lambda[j]$. For all $i$, the two vertices touched by $e_i$ are thus $v_{f(i)}$ and $v_i$.

(2) For $i$ from $m$ to 1:
If $v_i$ is a leaf, set $W_i' := 1$ and $W_i := l_{e_i}$.
Otherwise, set $x := \prod_{j\in\Lambda[i]} W_j$; set $W_i' := (\sum_{j\in\Lambda[i]} W_j'/W_j)x$; set $W_i := l_{e_i}W_i' + x$.

(3) For $i$ from 1 to $m$:
Set $x := \prod_{j\in\Lambda[f(i)], j\neq i} W_j$ and $y := \sum_{j\in\Lambda[f(i)], j\neq i} W_j'/W_j$.
If $f(i) \neq 0$, set $x := xV_{f(i)}$ and $y := y + V_{f(i)}'/V_{f(i)}$.
Set $V_i' := xy$ and $V_i := l_{e_i}V_i' + x$.

Using Lemma 6, it can be shown inductively that for $i$ from 1 to $m$, $W_i$ is the weight of the subtree hanging from $v_{f(i)}$ by edge $e_i$, and $V_i$ is the weight of the subtree hanging from $v_i$ by edge $e_i$. Notice that if $i$ is the index of a leaf, then $V_i$ is the weight $w_T$ of the tree.

The search of step (1) can be executed in $O(|E|)$ time. Since the $\Lambda[i]$ produced are disjoint, $O(|E|)$ time is required for steps (2) and (3). A tree, each of whose internal vertices has degree at least three, has at most $2|L| - 3$ edges, so the algorithm requires $O(|L|)$ time.

LEMMA 8. *Given any $p \in P$, let $T_1, \cdots, T_n$ be the subtrees of $T$ that hang from the path $p$. Then $w_p = l_p \prod_{i=1}^{n} w_{T_i}$.*

*Proof.* By Lemma 4, $w_p = b_p l_p$. A $p$-elimination can be seen as the union of dissections of the trees $T_1, \cdots, T_n$ hanging from path $p$ (Fig. 3). The lemma follows from the definitions of $b_p$ and $w_T$.

THEOREM 3. *Given a tree $T$, all of whose internal vertices have degree at least three, the weights $w_p$ for all $p \in P$ can be computed in $O(|L|^2) = O(|P|)$ time. Equation* (1) *is solvable in $O(|P|)$ time.*

*Proof.* We assume the weights $W_i$ and $V_i$ hanging by all edges $e_i$ of $T$ have been precomputed using the algorithm described in the proof of Lemma 7. Consider the following recursive algorithm for finding the weights of all pairs that include a specific leaf $v_0$:

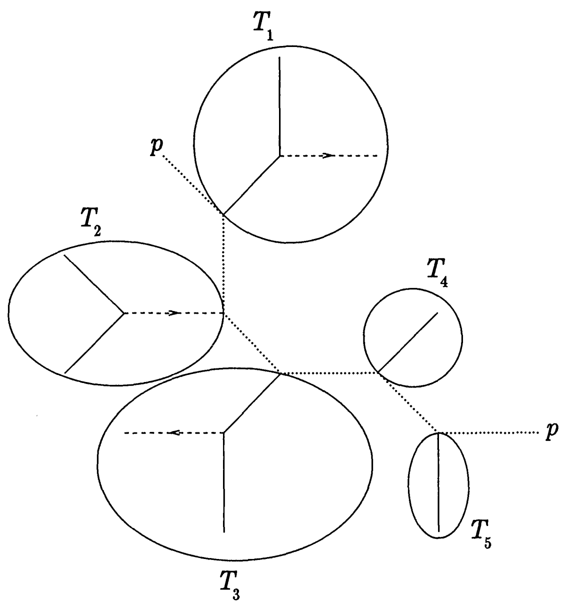


FIG. 3

(i) Mark all edges of the tree $T$. Execute subroutine A$(0, 1, v_0)$.

Subroutine A$(\sigma, \pi, v)$

(1) Set $\pi$ to $\pi$ times the weights of all subtrees hanging from $v$.

(2) If $v$ is a leaf other than $v_0$, record $\sigma\pi$ as the weight of the path from $v_0$ to $v$.

(3) Otherwise, for each marked edge $e_i$ adjacent to $v$ (and leading to vertex $v'$), unmark $e_i$ and execute subroutine $A(\sigma + l_e, \pi/W_iV_i, v')$.

The variable $\sigma$ is a running sum of the length of a path beginning at $v_0$. The variable $\pi$ is a running product of the weights of all subtrees hanging from the path under consideration. By Lemma 8, $\sigma\pi$ is the weight of the path from $v_0$ to $v$.

Since subroutine A is executed once for each vertex of the tree, and each edge is adjacent to two vertices, the total time required for the algorithm is $O(|E|) = O(|L|)$. By executing the algorithm for each leaf of the tree, the weights $w_p$ of all pairs are computed in $O(|L|^2) = O(|P|)$ time. (Each weight is calculated twice.) The precomputation of subtree weights takes time only $O(|L|)$. Since the precomputation yields $w_T$, by using the formula of Theorem 2 we can solve equation (1) in $O(|P|)$ time.

Note that for an arbitrary tree, matrix equation (1) is left unaltered by consolidating two lone edges that meet at a vertex and adding their lengths. Any tree can thus be transformed into an equivalent tree all of whose internal vertices have degree at least three. Thus given any tree, equation (1) is solvable in $O(|E|) + O(|P|)$ time.

**Conclusion.** While it is possible to solve (1) for $\vec{x}$ by matrix inversion, the graph theoretical solution described above is much more efficient. Furthermore, the formula for $\vec{x}$ given in Theorem 2 provides certain insights into the nature of the solutions. Among these is the fact that for positive edge lengths, all components of $\vec{x}$ are positive.

**Acknowledgment.** The author thanks Dr. David Lipman for suggesting the problem and for many helpful conversations.

## REFERENCES

[1] S. F. ALTSCHUL, *Gap costs for multiple sequence alignment*, J. Theoret. Biol., (1989), to appear.

[2] S. F. ALTSCHUL, R. J. CARROLL, AND D. J. LIPMAN, *Weights for data related by a tree*, J. Molec. Biol., (1989), to appear.

[3] S. F. ALTSCHUL AND D. J. LIPMAN, *Trees, stars, and multiple biological sequence alignment*, SIAM J. Appl. Math., 49 (1989), pp. 197–209.

[4] H. CARRILLO AND D. LIPMAN, *The multiple sequence alignment problem in biology*, SIAM J. Appl. Math., 48 (1988), pp. 1073–1082.

[5] J. FELSENSTEIN, *Maximum-likelihood estimation of evolutionary trees from continuous characters*, Am. J. Hum. Genet., 25 (1973), pp. 471–492.

[6] ———, *Numerical methods for inferring evolutionary trees*, Q. Rev. Biol., 57 (1982), pp. 379–404.

[7] S. B. NEEDLEMAN AND C. D. WUNSCH, *A general method applicable to the search for similarities in the amino acid sequences of two proteins*, J. Molec. Biol., 48 (1970), pp. 443–453.

[8] D. SANKOFF, *Minimal mutation trees of sequences*, SIAM J. Appl. Math., 28 (1975), pp. 35–42.

[9] D. SANKOFF AND R. J. CEDERGREN, *Simultaneous comparison of three or more sequences related by a tree*, In Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, D. Sankoff and J. B. Kruskal, eds., Addison-Wesley, Reading, MA, 1983, pp. 253–263.

[10] P. H. SELLERS, *On the theory and computation of evolutionary distances*, SIAM J. Appl. Math., 26 (1974), pp. 787–793.

[11] M. S. WATERMAN, *General methods of sequence comparison*, Bull. Math. Biol., 46 (1984), pp. 473–500.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 300-306, August 1989

# ON THE INVERTIBILITY OF THE DISCRETE RADON TRANSFORM*

GREGORY M. CONSTANTINE†

**Abstract.** The Radon transform is a useful device for analyzing multidimensional data. It is closely connected to what has become known as "projection pursuit." For the case of discrete data, theorems that address its invertibility are proven. Connections to the projective group over $GF(2)$ and block designs naturally arise. An extension of the Radon transform to joint densities is then investigated.

**Key words.** exploratory data analysis, projective group, block designs, projection pursuit

**AMS(MOS) subject classifications.** 62K10, 05B05

**1. Introduction.** Making sense out of multivariate discrete data often involves projecting the data onto a lower dimensional subspace and exploring it there for interesting effects. The subspace to project on is selected at will by the user. Some subspaces are of course more interesting than others, and search algorithms for relevant subspaces have been suggested by Huber (1981), and Friedman and Tukey (1979).

Projection pursuit for discrete data has been discussed in Diaconis (1983). This work extends several results presented in Diaconis' paper. The main focus is on understanding the conditions that make the Radon transform on discrete sets invertible. Results of this kind appear in Kung (1979) and Diaconis and Graham (1983). The former examines the independent sets of a matroid, while the latter relates essentially to binary codes. We give two large families of sets that admit invertible Radon transforms: the nonsingular linear maps of a finite dimensional vector space over $GF(2)$ (the field with two elements) and certain classes of block designs. Such is the case of BIB designs (referred to in the following as 2-designs), known to admit invertible transforms by Diaconis (1983) and the unpublished works cited there.

Most work on Radon transforms involves the recovery of a density as a function of one variable from its transform. A natural extension is to consider the recapture of a joint density in several variables. The transforms are defined as straightforward extensions of the univariate case. Inversion becomes a trickier business, since higher dimensions allow much more freedom of movement. Yet known structures, such as $t$-designs, are proved to admit invertible Radon transforms that recapture densities in up to $t/2$ variables. Information on construction of $t$-designs may be found in Constantine (1987, Chap. 7). Additional useful readings appear in Cameron and van Lint (1980) and Lander (1982).

**2. Recapturing a univariate density.**

**2.1.** Let $f$ be a function on a finite set $P$. We call the elements of $P$ *points* and denote by $|P|$ the cardinality of $P$. Let $B$ be a set of nonempty subsets of $P$. The Radon transform of $f$ at a set $\alpha \in B$ is

$$\bar{f}(\alpha) = \sum_{x \in \alpha} f(x).$$

The general problem that we address is this: characterize $B$ such that the set $\{\bar{f}(\alpha) : \alpha \in B\}$ of Radon transforms determines $f$ uniquely.

**2.2.** We shall write $v$ for $|P|$ and $b$ for $|B|$. Denote by $F$ the $v \times 1$ vector of values $f(x)$, as $x$ runs over $P$ in some fixed order. Similarly, let $\bar{F}$ be the $b \times 1$ vector of $\bar{f}(\alpha)$

* Received by the editors June 13, 1988; accepted for publication January 20, 1989.

† Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, Pennsylvania 15260.

as $\alpha$ runs over $B$. Furthermore, denote by $N$ the $v \times b$ incidence matrix of points versus the sets in $B$; that is, $N = (n_{x\alpha})$, with $n_{x\alpha} = 1$ if $x \in \alpha$ and $n_{x\alpha} = 0$ otherwise. With such notation we have

$$N^t F = \bar{F}. \tag{1}$$

We wish to solve for $F$. The solution is unique if and only if $N$ is of full row rank, in which case

$$F = (NN^t)^{-1} N\bar{F}. \tag{2}$$

We summarize in the following lemma.

LEMMA 1. *The Radon transform is invertible if and only if the point versus set incidence matrix is of full row rank over the field of rational numbers.*

Obviously, $N$ is of full row rank if and only if $NN^t$ is nonsingular. An obvious implication reminds us of Fisher's inequality in block designs: the Radon transform is invertible only if $b \geqq v$; that is, only if there are at least as many sets as points.

We would prefer a set-theoretic or geometric answer to our problem instead of the algebraic one that the lemma offers.

**2.3.** The matrix $N$ is of full row rank if and only if there exists a $v \times v$ submatrix $S$ of $N$ that is nonsingular over the rational numbers. Then $|S|$, the determinant of $S$, is a nonzero integer. It is clear that $|S|$ is odd if and only if $S$ is a nonsingular linear map of a vector space of dimension $v$ over $GF(2)$. This simple observation allows construction of many invertible Radon transforms, since the geometry of the group of such transformations is well understood. We summarize in the following theorem.

THEOREM 1. *If there exist $v$ columns in $N$ that form a basis over $GF(2)$, then the Radon transform is invertible.*

The matrix $S$ can of course have nonzero even determinant over the rational numbers. In such a case the transform is also invertible. A characterization of such matrices appears to be more difficult to obtain.

**2.4.** Denote the points of $P$ by $1, 2, \cdots, v$. The sum

$$\sum_{i \in \alpha} |\alpha|, \tag{3}$$

over all $\alpha$ in $B$ that contain point $i$, is called the *weight* of point $i$. Throughout this section we restrict attention to collections of sets $B$ that assign *equal weights* to all points. (For example, the points will have equal weights if all sets in $B$ have cardinality $k$ and if each point belongs to $r$ sets.) The common value $\sum_{i \in \alpha} |\alpha|$ is denoted by $d$.

Form the matrix

$$C = dI - NN^t. \tag{4}$$

The matrix $C$ is nonnegative definite, with zero row and column sums. It follows that $C$ is singular having the vector $\mathbf{1}$ (with all entries 1) in the kernel. One of its eigenvalues is therefore zero. Let $0 = \mu_0 \leqq \mu_1 \leqq \cdots \leqq \mu_{v-1}$ be the eigenvalues of $C$. It is obvious from (4) that $C$ and $NN^t$ admit the same set of (orthogonal) eigenvectors. The matrix $NN^t$ is therefore positive definite if and only if the spectral radius of $C$ is less than $d$ (i.e., if and only if $\mu_{v-1} < d$). We shall now attempt to find a good upper bound for $\mu_{v-1}$ and obtain a sufficient condition on the invertibility of the Radon transform by insisting that the upper bound be less than $d$.

Given $n$ numbers $x_1 \leqq x_2 \leqq \cdots \leqq x_n$, we denote by $\mu$ their average and by $\sigma^2$ their variance, that is, $\mu = n^{-1} \sum x_i$ and

$$\sigma^2 = \frac{1}{n} \sum (x_i - \mu)^2.$$

We assert that

$$x_n \leqq \mu + \sigma\sqrt{n-1}, \tag{5}$$

with equality if and only if $x_1 = x_2 = \cdots = x_{n-1}$. (To see this we may assume $\mu = 0$. Then the $x_i$'s can be interpreted (up to sign) as distances of one-pound weights from 0. Now the problem is how to balance these weights at zero without changing their variance such that the largest distance (i.e., $x_n$) is as large as possible. The worst possible case is intuitive: it has $x_1 = \cdots = x_{n-1}$ and thus formula (5) emerges.)

In the case of our matrix $C$, we have $v - 1$ numbers $\mu_1 \leqq \cdots \leqq \mu_{v-1}$. Their average is $\mu = (v - 1)^{-1} \operatorname{Tr} C$ and their variance is

$$\sigma^2 = (v-1)^{-1}[\operatorname{Tr}(C^2) - (v-1)^{-1}(\operatorname{Tr} C)^2].$$

The symbol Tr denotes the trace. Straightforward calculations show that

$$\begin{aligned}
\operatorname{Tr} C &= dv - \operatorname{Tr}(NN^t),\\
\operatorname{Tr} C^2 &= d^2v - 2d\operatorname{Tr}(NN^t) + \operatorname{Tr}((NN^t)^2),\\
\mu &= (v-1)^{-1}[dv - \operatorname{Tr}(NN^t)],\\
\sigma^2 &= (v-1)^{-2}[(v-1)\operatorname{Tr}((NN^t)^2) - (\operatorname{Tr}(NN^t))^2 + 2d\operatorname{Tr}(NN^t) - d^2v].
\end{aligned}$$

These calculations and the bound in (5) yield the following theorem.

THEOREM 2. *If each of the $v$ points has weight $d$, and if*

$$\begin{aligned}
\mu + \sigma\sqrt{v-2} = {} & (v-1)^{-1}[dv - \operatorname{Tr}(NN^t)]\\
& + (v-2)^{1/2}(v-1)^{-1}[(v-1)\operatorname{Tr}((NN^t)^2) - (\operatorname{Tr}(NN^t))^2\\
& + 2d\operatorname{Tr}(NN^t) - d^2v]^{1/2} < d,
\end{aligned}$$

*then the Radon transform is invertible.*

The usefulness of Theorem 2 lies in providing a sufficient condition for invertibility only in terms of traces of $NN^t$ and $(NN^t)^2$. Both traces are easy to compute. The next section looks at several special cases involving well-known block designs.

From Theorem 2 it easily follows that if $B$ consists of blocks of a BIBD (or 2-design), then the Radon transform is invertible. This result is proved in Diaconis (1983), and in some of the references that he cites. Indeed, $\mu = (v - 1)^{-1}vr(k - 1)$ and $\sigma^2 = 0$ for a BIBD, while $d = rk$. The inequality $\mu < rk$ is easily verified (since $k < v$).

**2.5.** Suppose $B$ is such that each set has $k$ points and each point is in $r$ sets. Then $\operatorname{Tr}(NN^t) = vr$, and Theorem 2 takes a simpler form, as stated in the following corollary.

COROLLARY 1. *If $(P, B)$ is a block design with $k$ points per block and $r$ blocks per point, and if the sum of squares of the entries of $NN^t$ is (strictly) less than*

$$(v-1)(v-2)^{-1}[rk - (v-1)^{-1}vr(k-1)]^2 + (v-1)^{-1}vr^2(v-2k+k^2),$$

*then the Radon transform is invertible.*

*Proof.* In this case $d = rk$, $\operatorname{Tr}(NN^t) = vr$, $\mu = (v - 1)^{-1}vr(k - 1)$, and

$$\sigma^2 = (v-1)^{-2}[(v-1)\operatorname{Tr}((NN^t)^2) - vr^2(v-2k+k^2)].$$

A straightforward, but possibly lengthy, calculation shows that the condition on invertibility written in Theorem 2 reduces to what appears in our corollary. This ends the proof.

Many designs that are used in statistical experiments are partial designs (known also as PBIB designs). They are based on association schemes. We refer to [Raghavarao, Chap. 8] or [Constantine, p. 297] for an introduction to association schemes. Let $A_i$ be the adjacency matrix of the $i$th associates in a scheme with $n$ classes. Denote by $v_i$ the number of $i$th associates of a (any) point; $\sum v_i = v$. We call $(P, B)$ a *partial design* if any pair of $i$th associates occurs in $\lambda_i$ blocks (i.e., elements of $B$). For a partial design the matrix $NN^t$ is therefore

$$NN^t = \sum_{i=1}^{n} \lambda_i A_i.$$

Fundamental properties of the scheme imply

$$\operatorname{Tr} A_i^2 = vv_i, \quad \text{and} \quad \operatorname{Tr} A_iA_j = 0, \quad \text{for } i \neq j.$$

Consequently, the sum of squares of entries in $NN^t$ is

$$\operatorname{Tr}((NN^t)^2) = \operatorname{Tr}((\sum \lambda_i A_i)^2) = \sum_{i,j} \lambda_i\lambda_j \operatorname{Tr} A_iA_j$$
$$= \sum vv_i\lambda_i^2 = v \sum v_i\lambda_i^2.$$

Corollary 1 now yields Corollary 2.

COROLLARY 2. *If $(P, B)$ is a partial design with $n$ associate classes whose parameters $v_i$ and $\lambda_i$ satisfy*

$$v \sum_{i=0}^{n} v_i\lambda_i^2 < (v-1)(v-2)^{-1}[rk-(v-1)^{-1}vr(k-1)]^2 + (v-1)^{-1}vr^2(v-2k+k^2),$$

*then the Radon transform is invertible.*

Certain classes of partial designs (e.g., Raghavarao (1971, p. 139)) can be shown to admit invertible transforms by the above corollary. Obviously, the transform is more likely to be invertible if the $\lambda_i$'s are nearly equal, that is, if they differ by at most 1.

A sharper result can be obtained for partial designs with two associate classes. In this case the Bose–Mesner algebra is generated by three elements: $I$ (the identity), $A_1$, and $A_2$ (the adjacency matrices of a strongly regular graph and its complement, respectively). The matrix $NN^t$ can be written as follows:

$$NN^t = rI + \lambda_1 A_1 + \lambda_2 A_2,$$

where $\lambda_i$ denotes the number of blocks containing a pair of $i$th associates. Matrices $A_1$ and $A_2$ share the same eigenvectors. Each has three distinct eigenvalues.

An eigenvector of $NN^t$ is the vector $\mathbf{1}$. If $NN^t$ is singular then for some nonzero vector $z$ orthogonal to $\mathbf{1}$ we have

$$(6) \qquad 0 = NN^t z = rIz + \lambda_1 A_1 z + \lambda_2 A_2 z.$$

Properties of the Bose–Mesner algebra yield that

$$A_2 z = -z - A_1 z.$$

Substituting in (6) we obtain:

$$(7) \qquad (\lambda_2 - \lambda_1) A_1 z = (r - \lambda_2) z.$$

This last equation tells us that $z$ is in the kernel of $NN^t$ if and only if $z$ is an eigenvector of $A_1$ with eigenvalue $(r - \lambda_2)(\lambda_2 - \lambda_1)^{-1}$.

$A_1$ is the adjacency matrix of a strongly regular graph with parameters $(v, v_1, a, c)$. Here $v$ is the number of points (or vertices), $v_1$ is the degree of any vertex, $a$ is the number of vertices adjacent to a pair of adjacent vertices, and $c$ is the number of vertices adjacent to a pair of nonadjacent vertices. The two nontrivial eigenvalues of $A_1$ are known to be

$$\mu_- = \tfrac{1}{2}\{(a-c) - [(a-c)^2 + 4(v_1 - c)]^{1/2}\}$$

and

$$\mu_+ = \tfrac{1}{2}\{(a-c) + [(a-c)^2 + 4(v_1 - c)]^{1/2}\}.$$

Equation (7) therefore implies that $NN^t$ is singular if and only if either

$$(r-\lambda_2)(\lambda_2-\lambda_1)^{-1} = \mu_- \quad \text{or} \quad (r-\lambda_2)(\lambda_2-\lambda_1)^{-1} = \mu_+.$$

The former can in fact only occur if $\lambda_2 < \lambda_1$, since $\mu_-$ is negative, while the latter can only occur if $\lambda_2 > \lambda_1$. This can be summarized as follows.

THEOREM 3. *Let $(P, B)$ be a partial design with two associate classes and design parameters $v_1, r, \lambda_1, \lambda_2$. Let the strongly regular graph of first associates have parameters $(v, v_1, a, c)$. Then the Radon transform with sets $B$ is singular if and only if either*

$$(r-\lambda_2)(\lambda_2-\lambda_1)^{-1} = \tfrac{1}{2}\{(a-c) - [(a-c)^2 + 4(v_1 - c)]^{1/2}\},$$

*or*

$$(r-\lambda_2)(\lambda_2-\lambda_1)^{-1} = \tfrac{1}{2}\{(a-c) + [(a-c)^2 + 4(v_1 - c)]^{1/2}\}.$$

Unlike for the case of BIBDs, there are partial designs that do not have an invertible Radon transform. The above result characterizes such situations when the scheme has two classes. Invertibility can be guaranteed, however, for large values of design parameters. As Theorem 3 shows, by keeping $|\lambda_2 - \lambda_1|$ small, a sufficiently large value of $r - \lambda_2$ ensures invertibility, since the right-hand sides in the two equations depend only on the parameters of the scheme and not of the design.

By specializing to various schemes, many known families of partial designs can be shown to have invertible transforms. We shall not do this here, contenting ourselves with one short example.

*Example.* The well-known Desargue configuration is a partial design based on a triangular association scheme with two classes. Vertices of the scheme are subsets of two elements of the set $\{1, 2, 3, 4, 5\}$. Two vertices are first associates if the two subsets are disjoint. The graph of first associates is the Petersen graph. Its parameters are $v = 10$, $v_1 = 3$, $a = 0$, and $c = 1$. Lines of the Desargue configuration correspond to cocliques in the Petersen graph. There are ten points and ten lines with three points per line. The design parameters are $v = 10$, $r = 3$, $\lambda_1 = 0$, and $\lambda_2 = 1$. Theorem 3 informs us that the Radon transform, whose sets are the lines of the Desargue configuration, is invertible.

We refer the interested reader to Chapter 8 of Raghavarao (1971) for a list of many families of partial designs.

**2.6.** Moments of the transform are related to moments of the original function. By assuming a uniform distribution over the values of $f$, and likewise for $\bar{f}$, we proceed in relating the means and variances of $f$ and $\bar{f}$. With some abuse of notation, and by using traces, it is straightforward to establish the following formulae:

(8) $\quad \operatorname{mean} \bar{f} = b^{-1}\mathbf{1}^t N^t f = \pm b^{-1}(f^t N J N^t f)^{1/2}, \quad$ and $\quad \operatorname{var} \bar{f} = b^{-1} f^t N(I - b^{-1}J)N^t f.$

Here $\mathbf{1}$ is the vector with all entries 1, $J = \mathbf{1}\mathbf{1}^t$, and $I$ is the identity matrix. For the case of a BIBD they reduce to those given in Theorem 2 of Diaconis (1983).

## 3. Retrieving a multivariate density.

**3.1.** A subset of $m$ points is called an $m$-subset. Let $f$ be a function defined on the $m$-subsets of $P$. In particular, $f$ may be a symmetric density function in $m$ variables. The Radon transform of $f$ at a subset $\alpha$ ($|\alpha| > m$) is

$$\bar{f}(\alpha) = \sum_{\sigma_m \subset \alpha} f(\sigma_m),$$

the sum extending over all $m$-subsets $\sigma_m$ of $\alpha$. Denote by $B$ the subsets $\alpha$ on which $\bar{f}$ is defined. We wish to find conditions on $B$ that make the transform $f \to \bar{f}$ invertible.

By a $t-(v, k, \lambda_t)$ design we understand a pair $(P, B)$ of points and $k$-subsets (called blocks) with the property that any $t$-subset is contained in $\lambda_t$ blocks. A $t-(v, k, \lambda_t)$ design is often called a $t$-design for short. It is known that a $t$-design is also an $i$-design, for $1 \leqq i \leqq t$, with

$$\lambda_i = \binom{v-i}{t-i}\binom{k-i}{t-i}^{-1} \lambda_t.$$

We denote by $N$ the $\binom{v}{s} \times \lambda_0$ incidence matrix of $s$-subsets of $P$ with the $\lambda_0$ blocks in $B$, where $(P, B)$ is a $2s$-design. It is a well-known result in the theory of $t$-designs that $N$ is of full row rank (see [Ray-Chaudhuri and Wilson, 1975]). We therefore have the following theorem.

THEOREM 4. *If the sets in $B$ form a $t$-design with $t$ even, then the Radon transform is invertible and retrieves any symmetric function in up to $t/2$ variables.*

The result can be generalized to functions on associate classes of the Johnson scheme. Write $NN^t = (\lambda_{\sigma\zeta})$, with $\lambda_{\sigma\zeta}$ the inner product of the rows of $N$ corresponding to $s$-subsets $\sigma$ and $\zeta$.

Let $B$ consist of sets, not necessarily of the same cardinality, with the property that $\lambda_{\sigma\zeta}$ depends only on the cardinality of the union $\sigma \cup \zeta$. We call such $B$ class compatible. Such structures exist, examples being the blocks of several $t$-designs of varying block sizes. By writing $\lambda_i$ for $\lambda_{\sigma\zeta}$ when $|\sigma \cup \zeta| = i$, we obtain

$$NN^t = \sum \lambda_i A_i,$$

where $A_i$ is the adjacency matrix of the $i$th associates in the Johnson scheme. The $\lambda_i$'s are called the class values. We refer to [Constantine, 1987] for notation and details. Let $y_j$ be the $j$th eigenvector of $A_i$ ($i = 1, \cdots, n$), where $n$ is the number of associate classes. The matrix $NN^t$ is singular if and only if for some $j$ we have

$$0 = \sum_i \lambda_i A_i y_j = \sum_i \lambda_i p(i,j) y_j = \left( \sum_i \lambda_i p(i,j) \right) y_j,$$

if and only if for some $j$

$$\sum_i \lambda_i p(i,j) = 0.$$

In the above writing $p(i,j)$ denotes the eigenvalue of $A_i$ associated to the eigenvector $y_j$. The eigenvalues of the Johnson's scheme are well known. They are the Eberlein polynomials

$$E(i,x)=\sum_{k=0}^{i}(-1)^k\binom{x}{k}\binom{n-x}{i-k}\binom{v-n-x}{i-k}, \qquad 0\leqq i\leqq n \tag{9}$$

evaluated at $x=j$. These observations may be summarized as follows.

THEOREM 5. *If $B$ is class compatible with class values $\lambda_i$, and if $\sum_{i=0}^{2s}\lambda_i E(i,j)\neq 0$, for all $j$, then the Radon transform is invertible and it recaptures a symmetric function in $s$ or fewer variables.*

## REFERENCES

P. CAMERON AND J. H. VAN LINT, *Graphs, Codes, and Designs*, London Mathematical Society Lecture Note Series 43, Cambridge University Press, Cambridge, 1980.

G. CONSTANTINE, *Combinatorial Theory and Statistical Design*, John Wiley, New York, 1987.

P. DIACONIS, *Projection pursuit for discrete data*, Stanford University Technical Report 198, Department of Statistics, Stanford, CA, 1983.

P. DIACONIS AND R. GRAHAM, *Finite Radon transforms on $Z_2^k$*, unpublished manuscript, 1983.

J. FRIEDMAN AND J. W. T. TUKEY, *A projection pursuit algorithm for exploratory data analysis*, IEEE Transactions on Computers, 9 (1974), pp. 881–890.

P. HUBER, *Projection pursuit*, Technical Report RJH-4, Department of Statistics, Harvard University, Cambridge, 1981.

J. KUNG, *The Radon transform of a combinatorial geometry I*, J. Combin. Theory Ser. A, (1979), pp. 97–102.

E. LANDER, *Symmetric Designs: An Algebraic Approach*, London Mathematical Society Lecture Note Series 74, Cambridge University Press, Cambridge, 1982.

D. RAGHAVARAO, *Constructions and Combinatorial Problems in Design of Experiments*, John Wiley, New York, 1971.

D. K. RAY-CHAUDHURI AND R. M. WILSON, *On t-design*, Osaka J. Math., 12 (1975), pp. 737–744.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 307-321, August 1989

# PERFORMANCE GUARANTEES ON A SWEEP-LINE HEURISTIC FOR COVERING RECTILINEAR POLYGONS WITH RECTANGLES*

D. S. FRANZBLAU†

**Abstract.** Finding the minimum number of rectangles required to cover a rectilinear or orthogonal polygon, where overlapping of rectangles is allowed, is one of several well-known, hard geometric decomposition problems. This paper reports the first results known that give worst-case performance bounds for an approximation algorithm for this problem. It is proved that partitioning the polygon into rectangles (with no overlapping) produces at most $2\theta + h - 1$ rectangles, where $\theta$ is the minimum number of rectangles in a cover, and $h$ is the number of holes. Examples are also given in which this bound is tight. The proof is based on counting arguments, and on Euler's formula for a planar graph. This paper shows that extending rectangles vertically and deleting duplicates produces at most $O(\theta \log \theta)$ rectangles. For the proof, geometric constraints are used to construct a large antirectangle, a set of points with no two contained in the same rectangle. This algorithm has an $O(n \log n)$ implementation using balanced trees, where $n$ is the number of vertices.

**Key words.** polygon, cover, rectangle, rectilinear, orthogonal, algorithm, approximation, heuristic, sweep-line

**AMS(MOS) subject classifications.** 05B40, 05B50, 52A45, 68Q25, 68R05

## 1. Introduction.

The problem of covering a rectilinear or orthogonal polygon with the minimum number of rectangles is one of a large class of geometric decomposition problems (several of which are surveyed in O'Rourke and Supowit (1983)). There has been little progress in finding algorithms for constructing minimum covers of arbitrary polygons with primitive shapes, thus, much attention has been focused on rectilinear geometry. The problem has applications to storing graphical images (Masek (1978)) and to the manufacture of integrated circuits (Mead and Conway (1980), Chaiken et al. (1981), Hegedüs (1982)). This and related problems have given rise to new classes of perfect graphs of interest in combinatorics (Saks (1982), Motwani, Raghunathan, and Saran (1988a), (1988b)).

The exact computational complexity of even this seemingly simple problem has remained open for a number of years. Masek (1978) proved that the problem is NP-hard for arbitrary polygons. Conn and O'Rourke (1987) showed that several restricted versions of the problem are also NP-hard. On the other hand, Franzblau and Kleitman (1984) gave a polynomial-time algorithm for covering vertically convex polygons with rectangles. They used a combinatorial duality theorem of Györi (1984), which improved an earlier result by Chaiken et al. (1981). Lubiw (1985), (1988a) showed that the problem was polynomial time for a somewhat larger class of polygons and has reported a weighted version of Györi's theorem (Lubiw (1988b)). Until now, the complexity of the problem for polygons with no holes was unknown. However, in a recent abstract, Culberson and Reckhow (1988) have reported that the problem is NP-hard even in this case.

The problem of covering a rectilinear polygon with the minimum number of "orthogonally convex" or "orthogonally star-shaped" components has a similar history (Keil (1986), Reckhow and Culberson (1987), Reckhow (1987), Culberson and Reckhow (1988), Motwani, Raghunathan, and Saran (1988a), (1988b)).

We distinguish the problem of *covering*, where overlapping of rectangles is allowed, from that of *partitioning* into disjoint rectangles. By contrast, the problem of partitioning an arbitrary rectilinear polygon into the minimum number of rectangles has an

* Received by the editors August 22, 1988; accepted for publication December 15, 1988.

† Department of Mathematics, Vassar College, Poughkeepsie, New York 12601.

$O(n^{5/2})$ solution (Ohtsuki (1982), Pagli et al. (1979)). When degenerate point holes are allowed, however, the problem is NP-hard (Lingas (1982)).

Since good algorithms for minimum cover problems are so scarce, it is worthwhile to look for approximation algorithms with guaranteed performance bounds. (See Garey and Johnson (1979, chap. 6) for an introduction to this approach.) This paper provides what we believe are the first results in this direction. Other covering heuristics have been reported, but with no analysis of performance bounds (Hegedüs (1982), Levcopoulos (1985)).

We first show that partitioning the polygon into rectangles using horizontal cuts produces at most twice the minimum number of rectangles if the polygon has no holes. More generally, partitioning always produces at most $2\theta + h - 1$ rectangles, where $\theta$ is the minimum number of rectangles in a cover, and $h$ is the number of holes in the polygon. We also give examples in which this bound is tight.

We then study a simple covering heuristic, called *Partition/Extend*, in which the rectangles in the partition are stretched vertically as far as possible, and duplicates are deleted. We show that this heuristic produces at most $O(\alpha \log \alpha)$ rectangles, where $\alpha$ is the size of an antirectangle, a set of points such that no two can be contained in the same rectangle. This gives a bound of $O(\theta \log \theta)$, since $\alpha \leqq \theta$. In our examples, the number of rectangles produced is at most $3\theta$, which we conjecture is the correct upper bound.

The organization of the paper is as follows. Section 2 contains basic definitions. The covering heuristic and main results are stated formally in § 3. An $O(n \log n)$ implementation of the heuristic is described in § 4. Some observations and open problems are given in § 5.

**2. Definitions.** A *rectilinear* or *orthogonal polygon* is a finite set of unit squares (or "pixels") on a two-dimensional, integer grid. That is, it is a simple polygon with integer vertices, edges aligned with the horizontal and vertical axes, and positive area. Such a polygon is also called a *polyomino*. In applications, all polygons are finite, so there is no loss of generality in assuming the polygon is embedded in a grid.

A *rectangle* will denote a rectangle (in the usual sense) which is a union of unit squares. That is, it is a rectangle aligned with the horizontal and vertical axes.

Given a rectilinear polygon $R$, a *rectangle cover* is a finite set of rectangles whose union is equal to $R$. Rectangles are allowed to overlap, but must be contained completely within $R$. A *minimum rectangle cover* is a rectangle cover containing the minimum possible number of rectangles. We denote the number of rectangles in a minimum cover by $\theta(R)$.

Dual to a rectangle cover is an *antirectangle* or *independent set*, a subset $A$ of unit squares in $R$, such that every rectangle inside $R$ contains at most one square of $A$. We denote the maximum number of squares in an antirectangle by $\alpha(R)$. It is easy to see that $\alpha(R) \leqq \theta(R)$. (See Fig. 1.) (Note that there are examples in which $\alpha < \theta$. See Chaiken et al. (1981), or Franzblau and Kleitman (1984, p. 166).)

Regarding $R$ as a closed planar region, the boundary of $R$ consists of horizontal and vertical line segments called *edges*. Edges intersect only at endpoints. A *vertex* of $R$ is the intersection of a horizontal and a vertical edge. There are three types of vertices, shown in Fig. 2. A *normal convex vertex* is the intersection of exactly two edges which form a 90° angle inside $R$. A *degenerate convex vertex* is the intersection of two pairs of edges forming two 90° angles. A *concave vertex* is the intersection of two edges forming a 270° angle.

We regard the vertices and edges of $R$ as a straight-line planar graph. A *hole* in $R$ is a bounded (finite area) face of this graph, contained in the complement of $R$. $R$ is

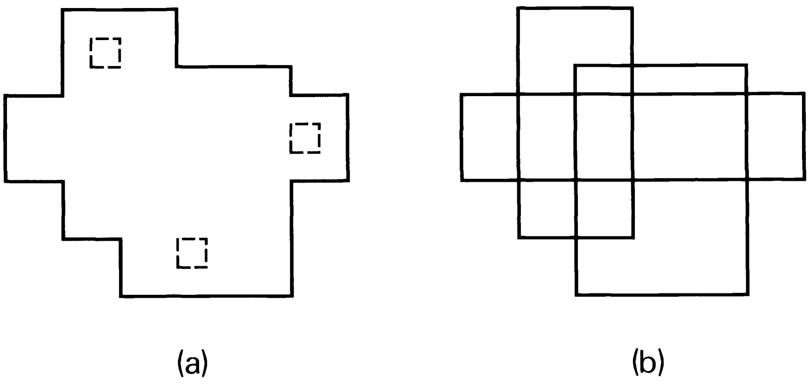


FIG. 1. (a) *Rectilinear polygon with antirectangle of size three (squares with dashed outlines).* (b) *Minimum rectangle cover, also of size three.*

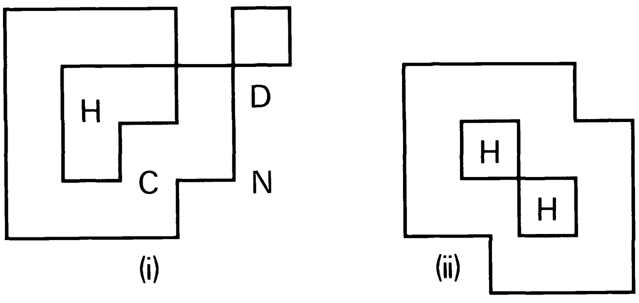


FIG. 2. *Polygon with two connected components,* (i) *and* (ii), *and three holes (H). Vertex N is normal convex, D is degenerate convex, and C is concave.*

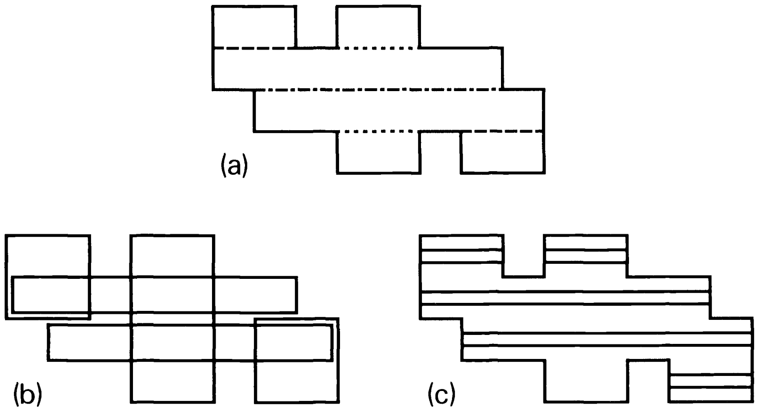


FIG. 3. (a) *Horizontal partition. Two type* 1 *chords* (– – –), *one type* 2 *chord* (–·–·), *and two type* 3 *chords* (· · ·) *are shown.* (b) *Rectangle cover determined by extending rectangles vertically. (Note that the two long horizontal rectangles are shown smaller than actual size for clarity.)* (c) *Pairs of parallel lines represent five inequivalent horizontal slices, which determine the same rectangle cover.*

*connected* if, after deleting the holes of $R$ (along with their boundaries), the remaining boundary is a connected graph. (See Fig. 2.)

A *horizontal chord* of $R$ is a horizontal line segment contained in $R$, such that at least one endpoint is a concave vertex, both endpoints lie on the boundary, and no other point lies on the boundary. (See Fig. 3(a).) Every concave vertex determines a unique horizontal chord. This notation is taken from Ohtsuki (1982).

**3. The covering heuristic and worst-case performance guarantees.** The covering heuristic can be described as follows.

ALGORITHM *Partition/Extend.*

*Input*: Rectilinear polygon $R$.
*Output*: Rectangle cover for $R$.

1. *Partition* $R$ into rectangles by cutting along each horizontal chord of $R$. (See Fig. 3(a).)
2. *Extend* each rectangle vertically inside $R$ until it is vertically maximal, i.e., touches the boundary on top and bottom. Delete any repeated rectangles. (See Fig. 3(b).)

Let $R$ be a rectilinear polygon, and let $p = p(R)$ be the number of rectangles obtained by horizontal partitioning alone (step 1). Let $\bar{p} = \bar{p}(R)$ be the number of rectangles obtained by Algorithm *Partition/Extend.*

Recall that $\theta = \theta(R)$ is the minimum number of rectangles in a cover, and $\alpha = \alpha(R)$ is the maximum number of unit squares in an antirectangle or independent set. Let $h = h(R)$ be the number of holes of $R$.

The main theorems of this paper are as follows.

THEOREM 1. *For any connected rectilinear polygon*, $p \leqq 2\theta + h - 1$.

THEOREM 2. *For any rectilinear polygon*, $\bar{p} \leqq 2\alpha(\log \alpha + 1)$.

From these theorems and the inequalities $\bar{p} \leqq p$ and $\alpha \leqq \theta$, we immediately derive the following performance bounds for Algorithm *Partition/Extend.*

COROLLARY 1. (Theorem 1.) *If $R$ has no holes, then* $\bar{p}/\theta \leqq 2$.

COROLLARY 2. (Theorem 2.) *For any rectilinear polygon*, $\bar{p}/\theta \leqq O(\log \theta)$.

Figures 4 and 5 give examples in which $p = 2\theta + h - 1$, so the bound of Theorem 1 is tight. In Fig. 4(a), $\bar{p} = p$, so the bound of Corollary 1 is tight.

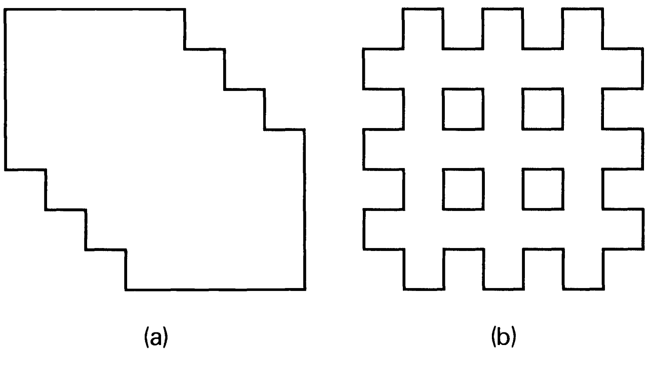


FIG. 4. *Two examples in which Theorem* 1 *is tight*: $p = 2\theta + h - 1$. (a) *Chain of $\theta$ rectangles* ($\theta = 4$). *There are no holes, and* $p = 2\theta - 1$. (b) *Lattice with* $\theta = 2k$ *rectangles* ($k = 3$). *The number of holes is* $(k-1)^2$, *and* $p = 2k + k^2 = 2\theta + h - 1$.

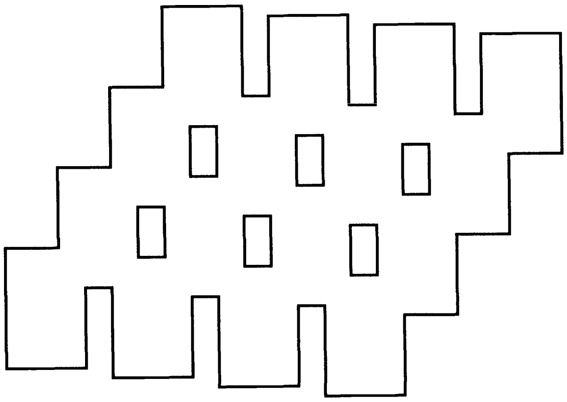


FIG. 5. *Another example in which Theorem* 1 *is tight. Formed from k rows of staircases of k non-overlapping rectangles each* ($k = 4$, $\theta = k^2$, $h = (k-1)(k-2)$). *The top staircase adds k rectangles to the partition. Every subsequent staircase adds* $3k - 1$ *rectangles. So* $p = k + (3k-1)(k-1) = 2\theta + h - 1 \leqq 3\theta$. *Extending rectangles vertically does not reduce the number.*

Figure 4(b) shows that the number of holes can be $\Theta(\theta^2)$, making the bound of Theorem 1 unsatisfactory in general. Note, however, that the heuristic gives the optimal solution in this case.

To prove Theorem 1, we find a relation between the minimum number of rectangles in a cover and the number of horizontal chords and vertices. We then relate the number of chords to the number of holes using Euler's formula for a planar graph.

There are three types of chords, illustrated in Figure 3(a). A *type* 1 *chord* contains exactly one concave vertex, a *type* 2 *chord* connects two concave vertices with opposite orientations, and a *type* 3 *chord* connects two concave vertices with the same orientation. Let $\mu_1$ and $\mu_2$ be the number of normal convex vertices and degenerate convex vertices, respectively. Let $c_1$, $c_2$, and $c_3$ be the number of type 1, type 2, and type 3 horizontal chords. Recall that $\theta$ is the number of rectangles in a minimum cover. We then have the following useful relation.

LEMMA 1.

$$(*) \qquad \mu_1 + 2\mu_2 + c_1 + 2c_2 \leqq 4\theta.$$

*Proof.* $4\theta$ is the number of corners of rectangles in a minimum cover. Each normal convex vertex of $R$ is also the corner of at least one covering rectangle. Each degenerate convex vertex yields two distinct corners. Given a type 1 chord, with concave vertex $u$ as an endpoint, there must be a rectangle covering $u$ whose corner lies on the chord. Using the same reasoning, each type 2 chord contains at least two rectangle corners. (See Fig. 6.) Since we have not counted any corners twice, the number of corners is at least $\mu_1 + 2\mu_2 + c_1 + 2c_2$. □

We can prove a similar relation involving $p$, the number of rectangles in the horizontal partition of $R$.

LEMMA 2.

$$(**) \qquad \mu_1 + 2\mu_2 + 3c_1 + 2c_2 + 2c_3 = 4p.$$

*Proof.* $4p$ is now the number of corners of rectangles in the *partition*. Each normal convex vertex is also a rectangle corner; each degenerate convex vertex yields two rectangle corners. Each type 1 chord determines three corners, while each type 2 or type 3 chord determines two corners. (See Fig. 7.) □

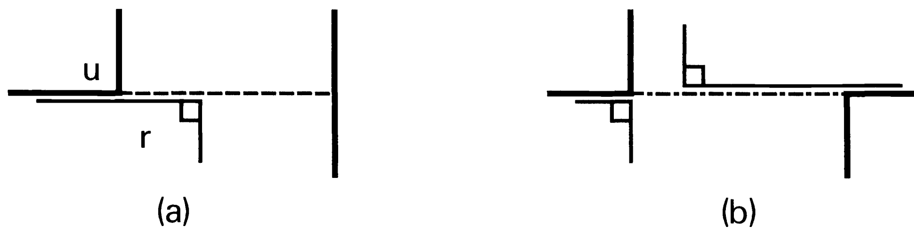


FIG. 6. *Illustration for the proof of Lemma* 1. (a) *Vertex u is the concave endpoint of a type* 1 *chord. There must exist a rectangle r covering u, with a corner on the chord.* (*Rectangle r is offset slightly from its actual position.*) (b) *Endpoints of a type* 2 *chord must be covered by two distinct rectangles with corners on the chord.* (*The two rectangles are offset slightly.*)

THEOREM 1. *For any connected rectilinear polygon*, $p \leqq 2\theta + h - 1$.

*Proof.* From (∗) and (∗∗) we obtain

$$(***) \qquad 2p \leqq 2\theta + (c_1 + c_3).$$

To relate $c_1$ and $c_3$ to the number of holes, we use Euler's formula for a connected planar graph, which says $v - e + f = 2$, where $v$ is the number of vertices, $e$ is the number of edges, and $f$ is the number of faces (including the unbounded face). (See Bondy and Murty (1976, p. 143).)

The vertices are the convex vertices of $R$, plus the endpoints of the chords. Thus, we have $v = \mu_1 + \mu_2 + 2(c_1 + c_2 + c_3)$.

The edges are the segments between vertices on the boundary of $R$, plus the chords. Each normal convex vertex is incident with two edges, each degenerate convex vertex with four edges, and every other vertex with three edges. Therefore, $2e = 2\mu_1 + 4\mu_2 + 6(c_1 + c_2 + c_3)$.

Finally, the number of faces is $f = p + h + 1$ where $p$ is the number of rectangles in the partition, and $h$ is the number of holes.

Using Euler's formula,

$$c_1 + c_3 = p + h - 1 - \mu_2 - c_2 \leqq p + h - 1.$$

Combining this with (∗∗∗) above yields $4p \leqq 4\theta + 2(p + h - 1)$, or $p \leqq 2\theta + h - 1$. □

The strategy of the proof of Theorem 2 is to show there is an antirectangle of size at least $(\bar{p}/2(\log \alpha + 1))$, where $\alpha$ is the maximum size of an antirectangle, and $\bar{p}$ is

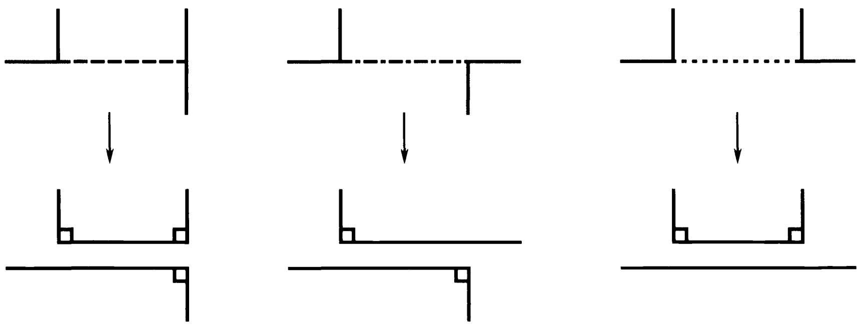


FIG. 7. *Illustration for the proof of Lemma* 2. *The top figures show the three types of chords. The bottom figures show the polygon* (*locally*) *after cutting along the chords. Each type* 1 *chord* (*left*) *yields three corners of rectangles in the partition, each type* 2 *chord* (*middle*) *or type* 3 *chord* (*right*) *yields two corners.*

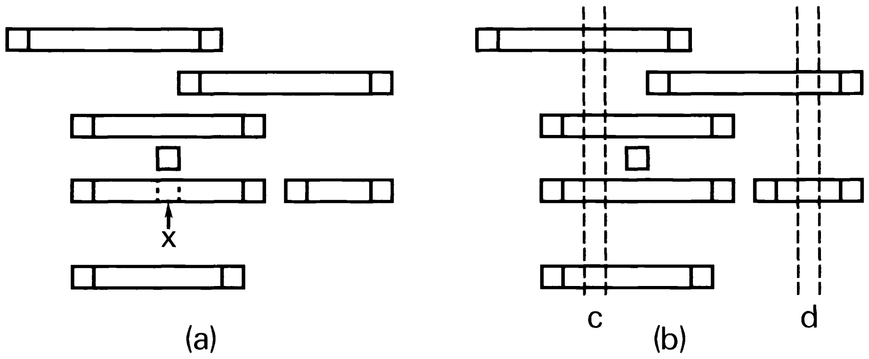


FIG. 8. *A set of inequivalent horizontal slices. The small squares are boundary squares.* (a) *Square x is blocked from above, but not from below.* (b) *Slices are separated by atoms c and d (pairs of vertical dashed lines).*

the number of rectangles in Algorithm *Partition/Extend*. The notation below will be used to describe the construction of the antirectangle.

A *horizontal slice* of $R$ is a horizontally maximal rectangle of unit height contained in $R$. Call a square touching the boundary of $R$, but not in $R$, a *boundary square*. A horizontal slice is a connected row of squares in $R$, delimited by two boundary squares.

Each horizontal slice determines a unique maximal rectangle in $R$, obtained by extending the slice vertically until it meets the boundary of $R$ on top and bottom. Two horizontal slices are *equivalent* if they determine the same maximal rectangle. A complete set of inequivalent horizontal slices determines a rectangle cover, since every square is contained in some slice. (See Figure 3(c).)

LEMMA 3. *The cover determined by a complete set of inequivalent horizontal slices is the same as the cover obtained by Algorithm Partition/Extend.*

*Proof.* The vertical sides of each rectangle in the horizontal partition must lie on the boundary of $R$. Thus, each rectangle of the partition contains a horizontal slice, so every rectangle generated by *Partition/Extend* is also generated by some slice. Conversely, every horizontal slice must be contained in some rectangle of the partition, so every rectangle generated by a slice is also generated by *Partition/Extend*. □

If $x$ is a unit square and $h$ is a horizontal slice, we say that $x$ is *blocked* from $h$ if there is no rectangle inside $R$ containing both $x$ and a square of $h$. This means that any rectangle containing $x$ and a square of $h$ must also contain a boundary square of $R$. Given a subset $S$ of horizontal slices and a unit square $x$, $x$ is *blocked from above* (with respect to $S$) if $x$ is blocked from every slice in $S$ (except the slice containing $x$) which is "above" $x$, i.e., whose vertical coordinate is greater than or equal to that of $x$. (See Fig. 8(a).) We define *blocked from below* similarly.

The following lemma shows that a set of blocked squares determines an antirectangle.

LEMMA 4. *Let $S$ be a subset of horizontal slices of a rectilinear polygon $R$. Let $A$ be a set of unit squares, such that each slice contains at most one square, and every square is blocked from above. Then $A$ is an antirectangle in $R$.*

*Proof.* Let $x_1$ and $x_2$ be two squares in $A$, and let $h_1$ and $h_2$ be the corresponding slices. We may assume that $h_2$ lies above $h_1$ (has the same or greater vertical coordinate). Since $x_1$ is blocked from above, no rectangle in $R$ contains both $x_1$ and $x_2$. □

By symmetry, the lemma also holds if "above" is replaced by "below."

Although it is not obvious how to construct a large set of blocked squares in general, if the set of slices can be "separated" by vertical lines, as described below, then we can

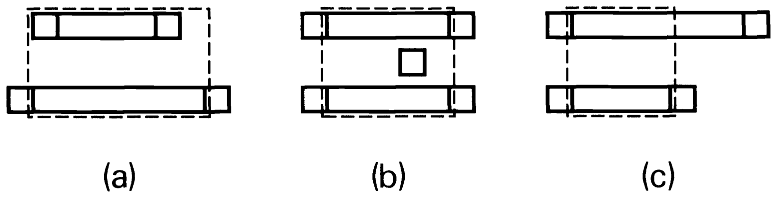


FIG. 9. *The three cases for rectangle $r_i$ (dashed outlines), defined in Lemma 5. (The small squares are boundary squares.) $I$ is the projection of the top slice on the x axis. $J$ is the projection of the bottom slice.* (a) $I \subset J$. *($r_i$ contains a boundary square.)* (b) $I = J$. *(Since $h_i$ and $h_{i-1}$ are inequivalent slices, there must be a boundary square between them, in $r_i$.)* (c) $J \subset I$. *($r_i$ may not contain any boundary squares.)*

show that either half the slices contain a square blocked from above or half contain a square blocked from below.

Call an infinite vertical strip of unit width an *atom*. A slice *contains* an atom $a$ if the slice intersects the vertical strip. Given a set of horizontal slices, $S$, a set of atoms $\{a_1, a_2, \cdots, a_k\}$ *separates* $S$ if every slice contains exactly one atom. (See Fig. 8(b).) (Note that not every set of slices has such a set of atoms; see Figure 3(c).)

LEMMA 5. *Let $S$ be a subset of inequivalent horizontal slices of a rectilinear polygon $R$. If atoms $\{a_1, a_2, \cdots, a_k\}$ separate $S$, then there is an antirectangle in $R$ containing at least $\lceil (|S| + k)/2 \rceil$ squares.*

*Remark.* It is possible to construct an example with $k = 1$ where this bound is tight.

*Proof.* For a given atom $a \in \{a_1, a_2, \cdots, a_k\}$, let $h_1, h_2, \cdots, h_m$ be the set of slices containing $a$, ordered from top to bottom.

Let $x_1$ be the intersection of atom $a$ and slice $h_1$. The square $x_1$ is blocked from above, since any slice $h$ above $h_1$ does not contain $a$, so that a rectangle containing $x_1$ and a square of $h$ must contain one of the boundary squares delimiting $h$.

For each slice $h_i$, $2 \leqq i \leqq m$, let $r_i$ be the smallest rectangle which contains the slice $h_i$ and extends to the row containing the slice $h_{i-1}$ above. (See Fig. 9.) If $r_i$ contains no boundary squares, then let $x_i$ be null. Otherwise, choose a boundary square $y$ in $r_i$ whose horizontal distance from atom $a$ is minimized. (Possibly, $y$ is contained in $a$.) Let $x_i$ be the square on $h_i$ directly below $y$. By the definition of $r_i$, $y$ lies in the same row or below that of $h_{i-1}$. (See Fig. 10.)

We claim that $x_i$ is blocked from above. If not, there is a slice $h$, above $h_i$, containing a square $x$, such that $x$ and $x_i$ are in the same rectangle. This slice $h$ must intersect the vertical strip containing $y$ and $x_i$. Since $y$ is in the same row or below $h_{i-1}$, $h$ cannot contain $a$. Thus, one of the boundary squares at the endpoint of $h$ lies in $r_i$ and is closer to $a$ than $y$, contradicting the choice of $y$. (See Fig. 10.)

For each of the $k$ atoms, construct the corresponding set of blocked squares, $\{x_1, x_2, \cdots, x_m\}$. Let $A$ be the union of these sets. $A$ is a set of squares satisfying the conditions of Lemma 4, and hence is an antirectangle. Using a symmetric construction, working from bottom to top at each atom, we can construct an antirectangle $B$ where each square is blocked from below.

The key observation is that, for any atom, if slice $h_i$ does not contain a square in $A$ then slice $h_{i-1}$ must contain a square in $B$. To see this, let $I$ and $J$ be the intervals on the horizontal axis which are the projections of $h_{i-1}$ and $h_i$, respectively. $I \cap J \neq \phi$ since $h_{i-1}$ and $h_i$ both contain $a$. The only case in which $h_i$ does not contain a square of $A$ is when $r_i$ contains no boundary squares. Note that if $J = I$, since there are no equivalent slices in $S$, there must be a boundary square between $h_{i-1}$ and $h_i$, in $r_i$. Thus, $h_i$ contains no square of $A$ only if $J$ is strictly contained in $I$. (See Fig. 9(c).) By inspection, $h_{i-1}$ must contain a square in $B$.

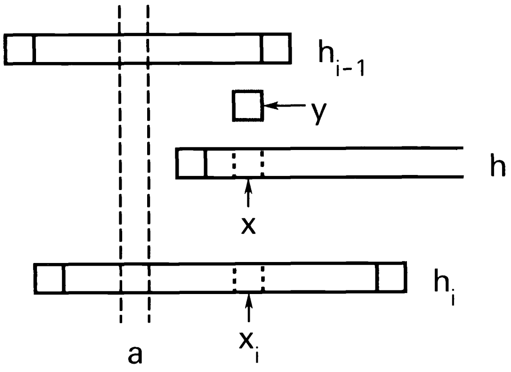


FIG. 10. *Definition of square $x_i$ in Lemma 5, and illustration of the proof that $x_i$ is blocked from above. Square y is the boundary square in rectangle $r_i$ closest to atom a (vertical dashed lines). If $x_i$ is not blocked from above, there is a slice h not containing atom a, and containing a square x, directly above $x_i$, as shown.*

Thus, the number of squares in $B$ is at least the number of "bottom" slices, i.e., the number of atoms plus the number of slices which do not contain a square of $A$. So, $|B| \geqq k + |S| - |A|$, or $|B| + |A| \geqq |S| + k$. The lemma follows from the pigeonhole principle. □

The construction of Lemma 5 cannot be used when a slice contains more than one atom. The next two lemmas will be used to show that we can extract a large separated subset in this case.

LEMMA 6. *Let $S$ be a set of slices, and let $\{a_1, a_2, \cdots, a_k\}$ be a set of atoms, sorted from left to right, such that each slice contains at least one atom. Then there is an $m$, with $m \leqq 1 + \log k$, such that $S$ can be partitioned into disjoint sets $S = S_1 \cup S_2 \cup \cdots \cup S_m$, where each $S_i$ is separated by a subset of the atoms.*

*Proof.* The proof is by induction on $k$. If $k = 1$ then $S$ is separated by the single atom $a$, so $m = 1$.

If $k > 1$, let $j = \lceil k/2 \rceil$ and let $S_1$ be the set of slices containing $a_j$. Trivially, $S_1$ is separated by $a_j$. Observe that $S$ is the disjoint union $S_L \cup S_1 \cup S_R$, where $S_L$ is the set of slices to the left of $a_j$, and $S_R$ is the set of slices to the right.

Since the atoms are sorted from left to right, $S_L$, with $\{a_1, a_2, \cdots, a_{j-1}\}$, and $S_R$, with $\{a_{j+1}, a_{j+2}, \cdots, a_k\}$, each satisfy the induction hypothesis. However, given a separated set in $S_L$ and a separated set in $S_R$, their union is also separated. If $S_L$ can be partitioned into $m_L$ separated sets, and $S_R$ can be partitioned into $m_R$ separated sets, then

$$m \leqq 1 + \max\{m_L, m_R\}.$$

By induction, $m_L \leqq 1 + \log(j-1) \leqq \log k$, and $m_R \leqq 1 + \log(k-j) \leqq \log k$. Therefore, $m \leqq 1 + \log k$. □

If $S$ is a set of slices, let $I(S)$ denote the set of (open) intervals obtained by projecting slices onto the horizontal axis.

LEMMA 7. *Let $S$ be a set of slices. Let $\delta$ be the maximum number of pairwise disjoint intervals in $I(S)$. Then, there is a set of $\delta$ atoms, such that each slice contains at least one atom.*

*Proof.* This is a well-known duality theorem for interval graphs. The lemma says that if $\delta$ is the size of a maximum independent set in the overlap graph determined by $I(S)$, then $\delta$ is also the size of a minimum cover by cliques. See Berge (1975, p. 9) for a proof. □

THEOREM 2. *For any rectilinear polygon*, $\bar{p} \leqq 2\alpha(\log \alpha + 1)$.

*Proof.* Let $S$ be the set of all inequivalent horizontal slices of $R$. If $\delta$ is as in Lemma 7, then there is a set of $\delta$ atoms such that each slice contains at least one atom. By Lemma 6, $S = S_1 \cup S_2 \cup \cdots \cup S_m$, where $m \leqq \log \delta + 1$, and each $S_i$ is separated by one or more atoms.

$|S| = \bar{p}$, by Lemma 3. From the pigeonhole principle, there is some $S_i$ with

$$|S_i| \geqq \frac{|S|}{\log \delta + 1} = \frac{\bar{p}}{\log \delta + 1}.$$

Using Lemma 5, we can construct an antirectangle from $S_i$, having size at least $\frac{1}{2}\bar{p}/(\log \delta + 1)$. Since $\alpha \geqq \delta$, the antirectangle has size at least $\frac{1}{2}\bar{p}/(\log \alpha + 1)$, or $\bar{p} \leqq 2\alpha(\log \alpha + 1)$. □

*Remark.* We can construct a set of $k = n(\log n + 1)$ intervals, with $n$ intervals of each length in $\{1, 2, 4, 8, \cdots\}$, such that the largest subset that is separated contains $2n - 1 \in O(\log k)$ intervals. Thus, this is essentially the best bound that can be proved using Lemma 5.

**4. Implementation of the heuristic.** To implement *Partition/Extend*, we first make a list of all "top corners" of rectangles in the horizontal partition. (See Fig. 11(b).) From this, it is easy to extract a list of all top edges of partition rectangles. (See Fig. 11(c)). To do this, we must find the corners which are not vertices of $R$ but are endpoints of (type 1) horizontal chords. We find the corners which are right endpoints of chords by running a vertical sweep-line from left to right across $R$ and updating the set of chords crossing the sweep-line at each step. We then perform the mirror-image procedure to find the vertices which are left endpoints. (The use of sweep-lines is a standard method in computational geometry; see, e.g., Preparata and Shamos (1985).)

Once we have a list of top edges of rectangles in the partition, we use a similar sweep-line technique to find the upper and lower edges of the vertically extended rectangles. We sweep from bottom to top to find the upper edges, then reverse direction to find the lower edges.

The $O(n \log n)$ time bound is based on the bound for sorting, and on the use of balanced trees. Here, $n$ is the number of vertices. If linked lists are used instead, the algorithm becomes $O(n^2)$. (Note that Ohtsuki (1982) reported the existence of an $O(n \log n)$ algorithm for horizontal partitioning but gave no details.) We give an informal presentation of further details and discussion of data structures below.

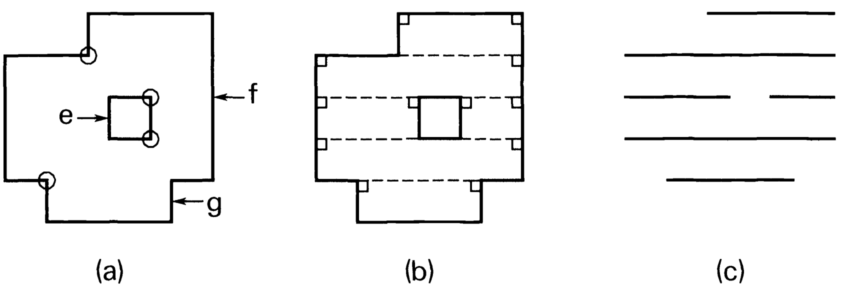


FIG. 11. (a) *Polygon with left concave corners circled, and right vertical edges labeled e, f, g.* (*Step* 1.1 *of implementation.*) (b) *Top corners of rectangles in the partition* (*marked with boxes*). *The dashed lines are horizontal chords.* (*Step* 2.) (c) *Top edges of rectangles in the partition.* (*Step* 3.)

ALGORITHM *Partition/Extend*.

*Input*: Rectilinear polygon $R$, represented as a list of vertices, sorted in order around each boundary component, oriented so that $R$ lies to the left. (If $R$ is connected, the outer boundary is oriented counterclockwise, and the hole boundaries are oriented clockwise.) (Note: any degenerate convex vertex will appear twice on the vertex list.)

*Output*: List of rectangles in a cover for $R$, each represented by a quadruple $[a, b, c, d]$, where $(a, b)$ is the projection of the rectangle on the $x$ (horizontal) axis, and $(c, d)$ is the projection on the $y$ (vertical) axis.

*Part* 1. **procedure** *Right endpoints*

{ *Left endpoints* is the mirror image of this procedure. }

*Output*: For each horizontal chord whose left endpoint is a concave vertex, output the right endpoint, $(x, y)$ (the intersection of the chord with the boundary of $R$).

{ Outline: To find the set of right endpoints, run a vertical sweep-line from left to right, stopping at each right vertical edge. At each stopping point, update the set of chords which extend rightward from some left concave vertex, across the sweep-line. Then, find all chords which intersect a right vertical edge on the sweep line and output their endpoints. }

*Step* 1.1. Making one pass over the vertex list of $R$, extract a list $V_L$, of all *left concave vertices*, and a list $E_R$, of all *right vertical edges* of $R$, each represented as a triple $[x, y_1, y_2]$, where $(y_1, y_2)$ is the projection on the $y$ axis. (See Fig. 11(a).)

> { $V_L$ represents the set of all chords whose left endpoint is concave. Any such chord must have its right endpoint on an edge in $E_R$. }

*Step* 1.2. Sort the edges in $E_R$, and the vertices in $V_L$ from left to right.

*Step* 1.3. Let $a$ be the minimum $x$-coordinate of the set of edges in $E_R$.

Let $e = [a, y_1, y_2]$, be an edge of $E_R$.

> Delete $e$ from $E_R$.
>
> Add to $L$ the set of vertices $(x, y)$ in $V_L$, with $x < a$. Delete these vertices from $V_L$.
>
> { $L$ represents the set of chords whose left endpoint lies strictly to the left of the line $x = a$ and whose right endpoint lies on or to the right of the line. }

*Step* 1.4. For each vertex $(x, y)$ in $L$, with $y_1 \leqq y \leqq y_2$,

{ there must be a chord with left endpoint $(x, y)$, which intersects edge $e$. }

> Delete $(x, y)$ from $L$;
>
> if $y_1 < y < y_2$, then add the point $(a, y)$ to the output list.

*Step* 1.5. Repeat Steps 1.3 and 1.4 until $E_R$ or $V_L$ is empty. □

*Part* 2. **procedure** *Top corners*

*Output*: Set of vertices which are "top corners" of rectangles in the horizontal partition. (See Fig. 11(b).)

Make one pass over the vertex list of $R$, examining incident edges, extracting all vertices which are either "upper convex vertices" or "lower concave vertices." To these, add the vertices found in Part 1. □

*Part* 3. **procedure** *Top edges*

*Output*: Set of top edges of rectangles in the partition, represented as triples $[x_1, x_2, y]$, where $(x_1, x_2)$ is the projection on the $x$ axis. (See Fig. 11(c).) Sort the list of top corners from Part 2, from top to bottom, then from left to right. Form top edges from adjacent pairs of vertices with the same $y$ coordinate. □

*Part* 4. **procedure** *Upper boundaries*

{ *Lower boundaries* is the mirror image of this procedure. }

*Output*: List of top edges of vertically maximal rectangles represented as triples $[x_1, x_2, d]$, where $(x_1, x_2)$ is the projection on the $x$ axis, and $d$ is the $y$-coordinate.

{ Note: when *Lower boundaries* is implemented, the input should be the list of triples $[a, b, d]$ from *Upper boundaries*. The output should be a list of quadruples $[a, b, c, d]$. }

{ Outline: use a sweep-line procedure, similar to that of *Right endpoints*, working from bottom to top, maintaining a list of all rectangles which extend upward to the current position of the sweep-line. }

```
Step 4.1. Let $T$ be the list of top edges, from Part 3.
          From the vertex list of $R$, extract a list $U$ of all "upper horizontal edges" of $R$.
Step 4.2. Sort $T$ and $U$ from bottom to top.
Step 4.3. Let $e$ be a horizontal edge $e = [x_1, x_2, d]$ in $U$ with minimum $y$ coordinate $d$.
              Delete $e$ from $U$.
          Add to $B$ all edges $[a, b, y]$ in $T$, with $y \leqq d$.
              Delete these edges from $T$.
              { $B$ contains all top edges of rectangles which can be extended upward at least to the line $y = d$. }
Step 4.4. For each edge $f = [a, b, c]$ in $B$, such that the (open) intervals $(a, b)$ and $(x_1, x_2)$ (the projection of edge $e$) intersect (in more than one point).
              Delete $f$ from $B$;
              Output the edge $[a, b, d]$.
Step 4.5. Repeat Steps 4.3 and 4.4 until $U$ or $T$ is empty. □
```

*Part* 5. **procedure** *Delete duplicate rectangles.*

Sort the list of rectangles lexicographically, and remove duplicates. □

Let $n$ be the number of vertices of $R$. Part 2 requires only $O(n)$ steps, while Parts 3 and 5 each require $O(n \log n)$ steps for sorting.

If $L$ is stored as a linked list, then Step 1.4 may require $\Theta(|L|)$ steps, yielding an $\Theta(n^2)$ algorithm. If $L$ is stored as a balanced tree (sorted by $y$ coordinate), all vertices in $L$ with $y_1 \leqq y \leqq y_2$ can be found in $O(\log |L| + k)$ steps, where $k$ is the number of vertices found. Each addition and deletion also requires only $O(\log |L|)$ steps. (For a discussion of balanced trees, see Knuth (1973, Chap. 6.2.3).) Step 1.4 thus requires a total of $O(n \log n)$ steps. Since $E_R$ and $V_L$ are sorted, Step 1.3 adds only $O(n)$ steps. In either case, $O(n)$ extra space is used.

The analysis of Part 4 is the same, with one exception. Instead of using a balanced tree to store $B$, use a *Group Tree* (due to McCreight, described in Ullman (1984, pp. 385–391)). (This data structure was introduced independently by Edelsbrunner, who called it an *Interval Tree*, described in Preparata and Shamos (1985, pp. 352–355).) Using this data structure, we can find all intervals $(a, b)$ which intersect the query interval $(x_1, x_2)$ in $O(\log n + k)$ steps, where $k$ is the number of intersections reported. Additions and deletions can also be performed in $O(\log n)$ steps each. Thus, the procedure can be performed in $O(n \log n)$ steps.

**5. Concluding remarks and open problems.** The main question raised by this work is whether there is any algorithm which always produces a cover with at most $c \cdot \theta$ rectangles, where $c$ is a constant. Note that a proof which uses the technique of constructing a large antirectangle, as in the proof of Theorem 2, will prove the stronger result that $\theta / \alpha$ is bounded.

We have shown that a simple, efficient heuristic for producing a rectangle cover yields a number of rectangles which is at worst twice the minimum if the polygon has no holes. We conjecture that the number of rectangles is within a constant factor of the minimum for an arbitrary rectilinear polygon, though we have not been able to prove this. The bound of $O(\theta \log \theta)$, from Theorem 2, is the same asymptotic bound obtained by Johnson (1974) on a heuristic for general set covering. His algorithm requires repeated searching for the maximal rectangle containing the most uncovered squares. The example of Fig. 4(a) has $\Theta(n^2)$ maximal rectangles, so a naïve implementation would use much more space and time than *Partition/Extend*.

There is some evidence that the upper bound for Algorithm *Partition/Extend* is $3\theta$. Observe that if there are no type 3 chords (as in Figs. 4(a) and 5), then the proof of Theorem 1 shows that $p \leqq 3\theta$. If no four vertices lie on the same horizontal line, then there can be no type 2 or type 3 chords, so $p \leqq 3\theta$ in this case also. On the other hand, the existence of many type 3 chords (or many vertices on the same line) may force rectangles to be deleted when rectangles are "extended," as in Figure 4(b).

There are many other algorithms which would be natural for further study. An obvious improvement to *Partition/Extend* is to use it both horizontally and vertically, then take the best cover. Another improvement is to find "degenerate" rectangles in the cover, those that are completely covered by a set of narrower rectangles, deleting them until stuck. (See Fig. 12.) In the example of Fig. 4(a), there is one degenerate rectangle. In Fig. 5, deleting degenerate rectangles yields an optimal cover.

Inserting or deleting one rectangle at a time, in a greedy manner, can yield covers with $\Omega(\theta^2)$ rectangles if done arbitrarily. However, there may be rules for selecting the next rectangle, with a much better bound (as in Johnson (1974)). A different type of heuristic was described informally by Hegedüs (1982). The idea is to first cover all "edge"

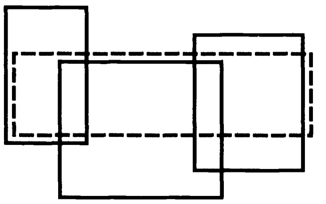


FIG. 12. "*Degenerate*" *rectangle* (*dashed outline, offset slightly*). *It is covered completely by the three rectangles with solid outlines.*

squares, greedily, obtaining a "reduced polygon," then repeating the process until all squares are covered. Note that finding a minimum cover for the set of edge squares is also NP-hard (Conn and O'Rourke (1987)).

Finally, there are several nonrectilinear covering problems that have practical applications, for which it would be worthwhile to study the performance of approximation algorithms. One problem is covering a (nonrectilinear) polygon with rectangles, allowing rotations (and assuming either no acute angles, or a means of approximating an acute angle with rectangles). This problem has been studied by Levcopoulos and Lingas (1984), and by Levcopoulos (1985). Allowing a discrete set of edge orientations, such as 45° and 90°, would also be of interest in this case. Another problem is covering an arbitrary polygon with convex polygons (proved NP-hard by O'Rourke and Supowit (1983)). There is a rather complicated polynomial-time algorithm for finding a minimum partition (when there are no holes), due to Chazelle and Dobkin (1979). It would be interesting to know whether a relation between covering and partitioning, similar to that of Theorem 1, holds in this case also.

## REFERENCES

C. BERGE (1975), *Perfect graphs*, Studies in Graph Theory, Part I, D. R. Fulkerson, ed., MAA Stud. Math., 11, pp. 1–22.

J. A. BONDY AND U. S. R. MURTY (1976), *Graph Theory with Applications*, North-Holland, Amsterdam.

S. CHAIKEN, D. J. KLEITMAN, M. SAKS, AND J. SHEARER (1981), *Covering regions by rectangles*, SIAM J. Algebraic Discrete Methods, 2, pp. 394–410.

B. M. CHAZELLE AND D. DOBKIN (1979), *Decomposing a polygon into its convex parts*, In Proc. 11th ACM Sympos. on Theory of Computing, (STOC 1979), pp. 33–48.

H. E. CONN AND J. O'ROURKE (1987), *Some restricted rectangle covering problems*, Johns Hopkins University, Baltimore, MD, submitted.

J. CULBERSON AND R. A. RECKHOW (1988), *Covering polygons is hard* (preliminary abstract), Proc. 29th IEEE Sympos. Found. of Comp. Sci. (FOC 1988), pp. 601–611.

D. S. FRANZBLAU AND D. J. KLEITMAN (1984), *An algorithm for covering polygons with rectangles*, Inform. and Control, 63, pp. 164–189. Extended abstract, *An algorithm for constructing regions with rectangles*, in Proc. 16th ACM Sympos. on Theory of Computing, (STOC 1984), pp. 167–174.

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

E. GYÖRI (1984), *A minimax theorem on intervals*, J. Combin. Theory Ser. B, 37, pp. 1–9.

A. HEGEDÜS (1982), *Algorithms for covering polygons by rectangles*, Comput. Aided Geom. Design, 14, pp. 257–260.

D. S. JOHNSON (1974), *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9, pp. 256–278.

J. M. KEIL (1986), *Minimally covering a horizontally convex orthogonal polygon*, in Proc. 2nd. ACM Symp. on Computational Geometry, pp. 43–51.

D. KNUTH (1973), *The Art of Computer Programming*, Vol. III, Addison-Wesley, Reading, MA.

C. LEVCOPOULOS (1985), *A fast heuristic for covering polygons by rectangles*, Tech. Rep. S-581 83, Linköping University, Linköping, Sweden.

C. LEVCOPOULOS AND A. LINGAS (1984), *Covering polygons with minimum number of rectangles*, in Lecture Notes in Comput. Sci. 166, Springer-Verlag, New York, Berlin, pp. 63–72.

A. LINGAS (1982), *The power of non-rectilinear holes*, 9th Int. Colloq. Autom. Lang. Prog., (ICALP 1982).

A. LUBIW (1985), Orderings and some combinatorial optimization problems with some geometric applications, Ph.D. Thesis, University of Toronto, Ontario, Canada.

——— (1988a), *The Boolean basis problem, and how to cover polygons by rectangles*, University of Waterloo, Ontario, Canada, in preparation.

——— (1988b), *A Weighted* Min Max *Relation for Intervals*, University of Waterloo, Ontario, Canada, unpublished manuscript.

W. J. MASEK (1978), *Some* NP-*complete set covering problems*, Massachusetts Institute of Technology, Cambridge, MA, unpublished manuscript. Referenced in Garey and Johnson (1979), p. 232.

C. MEAD AND L. CONWAY (1980), *Introduction to* VLSI *Systems*, Addison–Wesley, Reading, MA, Chaps. 2 and 4.

R. MOTWANI, A. RAGHUNATHAN, AND H. SARAN (1988a), *Perfect graphs and orthogonally convex covers*, University of California at Berkeley, submitted. Presented at 4th SIAM Conference on Discrete Mathematics, San Francisco, June 1988.

——— (1988b), *Covering orthogonal polygons with star polygons: The perfect graph approach*, in Proc. 4th ACM Symp. on Computational Geometry, pp. 211–223.

T. OHTSUKI (1982), *Minimum dissection of rectilinear regions*, Proc. 1982 IEEE Symp. on Circuits and Systems, Rome, pp. 1210–1213.

J. O'ROURKE AND K. J. SUPOWIT (1983), *Some NP-hard polygon decomposition problems*, IEEE Trans. Inform. Theory, 29, pp. 181–190.

L. PAGLI, E. LODI, F. LUCCIO, C. MUGNAI, AND W. LIPSKI (1979), *On two dimensional data organization* 2, Fund. Inform., 2, pp. 211–226.

F. P. PREPARATA AND M. I. SHAMOS (1985), *Computational Geometry*, Springer-Verlag, Berlin, New York.

R. A. RECKHOW (1987), *Covering orthogonally convex polygons with three orientations of dents*, Tech. Rep. TR 87-17, Dept. of Comp. Sci., University of Alberta, Edmonton, Canada.

R. A. RECKHOW AND J. CULBERSON (1987), *Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons*, Proc. 3rd. ACM Symp. on Computational Geometry, pp. 268–277; *Orthogonally convex coverings of orthogonal polygons without holes*, J. Comput. System Sci., to appear.

M. SAKS (1982), *A class of perfect graphs associated with planar rectilinear regions*, SIAM J. Algebraic Discrete Methods, 3, pp. 330–342.

J. D. ULLMAN (1984), *Computational Aspects of* VLSI, Computer Science Press, Rockville, MD.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 322-328, August 1989

# CONSTRUCTING A MAXIMAL INDEPENDENT SET IN PARALLEL*

MARK GOLDBERG†‡ AND THOMAS SPENCER†

**Abstract.** The problem of constructing in parallel a maximal independent set of a given graph is considered. A new deterministic NC-algorithm implemented in the EREW PRAM model is presented. On graphs with $n$ vertices and $m$ edges, it uses $O((n + m)/\log n)$ processors and runs in $O(\log^3 n)$ time. This reduces by a factor of $\log n$ both the running time and the processor count of the previously fastest deterministic algorithm that solves the problem using a linear number of processors.

**Key words.** parallel computation, NC, graph, maximal independent set, deterministic

**AMS(MOS) subject classification.** 68R10

**1. Introduction.** The problem of constructing in parallel a maximal independent set of a given graph, MIS, has been investigated in several recent papers. Karp and Wigderson proved in [KW] that the problem is in NC. Their algorithm finds a maximal independent set of an $n$-vertex graph in $O(\log^4 n)$ time and uses $O(n^3/\log^3 n)$ processors. In successive papers, the authors proposed algorithms which either are faster or use a smaller number of processors. Luby in [L1] and Alon, Babai, and Itai in [ABI] presented probabilistic algorithms running in $O(\log^2 n)$ time on a EREW PRAM with a linear number of processors. Luby also described a technique for converting probabilistic algorithms into deterministic ones; the technique preserves the running time but requires an increase in the number of processors used to $O(n^2m)$, where $m$ is the number of edges in the graph. The first deterministic NC-algorithm on a linear number of processors (EREW model) was constructed in [GS]; its running time is $O(\log^4 n)$. Recently, Luby [L2] proposed a general method for converting randomized parallel algorithms into deterministic ones, which does not require an increase in the number of processors. In the case of MIS, the method yields a new algorithm running on a linear number of processors in polylogarithmic time; however, it runs slower than that in [GS]. All NC-algorithms for MIS mentioned above use the following top-level design proposed in [KW]:

> Start with an empty independent set $I$. Find an independent set $I'$, add it to $I$, and delete $I'$ and the vertices adjacent to a vertex in $I'$ from the graph. Repeat the previous step until the graph is empty.

Call *FINDSET* the procedure which constructs $I'$. One can easily prove that an algorithm with such a structure belongs to NC if *FINDSET* is designed so that, on any $n$-vertex graph ($n > 0$), it runs in polylogarithmic time and delivers an independent set $C$ such that $|C \cup N(C)| = \Omega(n/\log^s n)$ for some fixed $s \geqq 0$; ($N(C)$ is the set of neighbors of $C$).

In this paper, we present a new deterministic algorithm for MIS which improves the running time of the algorithm in [GS] by a factor of $\log n$ and simultaneously reduces, also by a factor of $\log n$, the number of processors used. The algorithm is implemented in the EREW model of computation. Thus, the processor-time product of the algorithm in [GS] is improved by $\log^2 n$. These gains are due to the new version of the *FINDSET* procedure. The new procedure finds in $O(\log^2 n)$ time an independent set $I$ such that removing $I \cup N(I)$ reduces the size of the graph by half. Thus, the procedure *FINDSET* needs to be called only $O(\log n)$ times.

To reduce the number of processors used, we modify the definition of the size of the graph so that it takes into account the number of edges deleted. The result of this

* Received by the editors April 25, 1988; accepted for publication (in revised form) November 23, 1988.

† Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180.

‡ This work was supported in part by the National Science Foundation under grant DCR-8520872.

modification is that the processor-reduction technique of Miller and Reif [MR] can be applied to reduce the number of processors necessary by a factor of log $n$.

Both the algorithm of [GS] and the one of this paper use the idea of a *partial coloring*. A partial coloring is an assignment of colors to some, not necessarily all, of the vertices such that no two adjacent vertices have the same color. Examples of partial colorings include the trivial coloring where every vertex has a different color and the empty coloring where no vertex is colored. If a coloring assigns colors to all of the vertices, then it is called *complete*. The set of vertices with the same color is called a *color class*.

*FINDSET* constructs a sequence of partial colorings starting with the trivial coloring. The objective is to find a "big" color class $C$ (the definition of "big" is given later). If such a color class $C$ is found, *FINDSET* deletes $C \cup N(C)$ from the graph; if no color class is "big," some of the color classes are merged. A side effect of this is that some vertices become uncolored. *FINDSET* halts when all vertices are either deleted or uncolored. It returns the union of all the "big" color classes it found.

The success of such an approach depends on the definition of a "big" color class and on the efficiency of merging. When an algorithm decides to merge, it should do it "fast" and so that "not-too-many" vertices are decolored. The technical means by which this task is accomplished in our implementation is the comparison of two representations of the edge set $E$ of a colored graph. The first one views $E$ as the union of the sets $L(C)$ of edges with an endpoint in the color class $C$, where $C$ ranges over the set of all color classes. Roughly speaking, $L(C)$ measures the size of the part of the graph that would be deleted if $C$ is added to $I$. The second representation of $E$ partitions the edges into classes so that the membership of an edge is determined by the colors of both its endpoints. Every class of the second representation measures the size of the graph which would be decolored if merging were performed according to this particular class. Propositions 1 and 2 establish that if there are no big classes, then there is efficient merging.

We expect that the consideration of these representations can be helpful in the design of parallel algorithms for other graph problems.

**2. Terminology.** The definitions of class NC and models of computations can be found in [P], [V], and [KR]. The graph-theoretic terminology used in this paper is standard [BM]. The degree of a vertex $v$ in a subgraph $H$ is denoted $deg_H(v)$. Given a set $K \subset V(G)$ of vertices, $\sigma_H(K) = \sum_{v \in K}(1 + deg_H(v))$ is called the *weight* of $K$ in $H$. If the graph is understood, the indices in $deg_H$ and $\sigma_H(K)$ are omitted. We use *weight* as the definition of size when we say that *FINDSET* finds an independent set $I$ such that the deletion of $I \cup N(I)$ reduces the size of $H$ by half.

Let $\mathbf{C} = [C_0, \cdots, C_{r-1}]$ be the collection of color classes of a partial coloring $\phi$. Then, $L_i$ denotes the set of edges which have one endpoint in $C_i$, and $N_i$ denotes the set of vertices which have neighbors in $C_i$. The consideration of $L_i$'s yields the first method of classifying the edges. To understand the second method, we need to define the functions

$$\chi(p) = \begin{cases} p & \text{if } p \text{ is odd,} \\ p-1 & \text{if } p \text{ is even.} \end{cases}$$

$$rev(i,q;p) = (q-i) \bmod \chi(p).$$

$$index(i,j;p) = \begin{cases} (i+j) \bmod \chi(p) & \text{if } 0 \leqq i,j \leqq \chi(p)-1, \\ 2i \bmod \chi(p) & \text{if } j = \chi(p), \\ 2j \bmod \chi(p) & \text{if } i = \chi(p). \end{cases}$$

One can readily prove that for every $q$ and $i$ $(0 \leqq i, q \leqq \chi(p) - 1)$, $j = rev(i, q; p)$ is the only integer in the interval $[0, \chi(p) - 1]$ with $index(i, j; p) = q$.

Let $\phi$ be a vertex coloring and $\mathbf{C} = [C_0, \cdots, C_{r-1}]$ be the set of its color classes. For $k = 0, \cdots, \chi - 1$ let $\pi_k = \{(C_i, C_j)\colon index(i, j; r) = k\}$ be a partition of $\mathbf{C}$ into $\lfloor r/2 \rfloor$ pairs and possibly one singleton. These $\chi$ partitions of $\mathbf{C}$ are called the regular partitions of $\mathbf{C}$. One can readily check, that for all $i \neq j$, $\pi_i$ and $\pi_j$ do not share a common pair. Thus, the collection $\{\pi_0, \cdots, \pi_{\chi-1}\}$ presents a minimal edge coloring of the complete graph whose vertices are $C_0, \cdots, C_{r-1}$.

Fix $k$ and let $\pi = \pi_k$. For each $l = 0, \cdots, \lfloor r/2 \rfloor$, define $B_l = (C_{i_l} \cap N(C_{j_l})) \cup (C_{j_l} \cap N(C_{i_l}))$ and $B = \bigcup_{l=0}^{\lfloor r/2 \rfloor - 1} B_l$. The weight $\sigma(\pi)$ of $\pi$ is then given by $\sigma(\pi) = \sigma(B)$.

**3. An outline of the algorithm.** In this section, we present a description of *FINDSET* and prove that every application of *FINDSET* reduces the weight of the graph by half.

Let $G$ be a graph with $n$ vertices and $m$ edges. *FINDSET* starts by defining an empty independent set $I$ and a trivial vertex coloring on the vertices of $G$. Then, it proceeds in phases. At every phase, one of the following actions is performed:

$\langle 1 \rangle$ A color class $C^*$ is found for which $\sigma(N(C^*)) \geqq (n + m)/\log n$. All vertices of $C^*$ are added to $I$ and all vertices from $C^* \cup N(C^*)$ are deleted.

$\langle 2 \rangle$ The color classes are partitioned into pairs $(C_i, C_i')$, with possibly one color class left over. The two color classes of each pair are merged. To make the merged color class independent, the weights of the sets $C_i \cap N(C_i')$ and $C_i' \cap N(C_i)$ are compared and the vertices of the set with the smaller weight are decolored.

Action $\langle 1 \rangle$ is done whenever possible; action $\langle 2 \rangle$ is done only when there is no suitable color class $C^*$. The actions are executed until at most one color class is left. If it is indeed one color class, then independent of its weight the color class is added to $I$.

Action $\langle 2 \rangle$ is done by the procedure *HALF*. It constructs a regular partitioning $\pi$ of the minimal weight, and merges every pair of color classes matched by $\pi$.

The following propositions show that action $\langle 2 \rangle$ does not decolor too many vertices.

PROPOSITION 1. *Let* $\mathbf{C} = [C_0, \cdots, C_{r-1}]$ *be the set of color classes of a coloring* $\phi$ *and* $\{\pi_0, \cdots, \pi_{\chi-1}\}$ *be the set of regular partitionings of* $\mathbf{C}$. *Then*

$$(*) \qquad \sum_{j=0}^{\chi-1} \sigma(\pi_j) \leqq \sum_{i=0}^{r-1} \sigma(N(C_i)).$$

*Proof.* The set of partitionings $\{\pi_j\}$ $(j = 0, \cdots, \chi - 1)$ contains every pair of color classes exactly once. Therefore, for every colored vertex $v$, its contribution to the left part of the inequality is equal to $\sigma(v) \times$ (the number of color classes that contain vertices adjacent to $v$). Obviously, the contribution of $v$ to the right part is as big as that.

PROPOSITION 2. *Let* $\phi$ *be a coloring and* $\mathbf{C} = [C_0, \cdots, C_{r-1}]$ *be the set of its color classes. If for every* $i \geqq 0$, $\sigma(N(C_i)) \leqq (n + m)/\log n$, *then there is a regular partitioning* $\pi$ *of* $\mathbf{C}$ *and a set* $D$ *of vertices such that*

(1) *for every pair* $C', C''$ *of color classes matched by* $\pi$, $C' \cup C'' - D$ *is an independent set*;

(2) $$\sigma(D) \leqq \frac{n+m}{2 \log n} \frac{r}{\chi(r)}.$$

*Proof.* If for every $i \geqq 0$, $\sigma(N(C_i)) \leqq (n+m)/\log n$, then it follows from $(*)$ that

$$\sum_{j=0}^{x-1} \sigma(\pi_j) \leqq \frac{(n+m)}{\log n} r,$$

which in turn implies the existence of a regular partitioning $\pi_k$ of $\mathbf{C}$ with $\sigma(\pi_k) \leqq ((n+m)/\log n)(r/\chi(r))$.

Let $\{(C_{i_l}, C_{j_l})\}$ be the set of pairs of $\pi_k$. For every $l = 0, \cdots, \lfloor r/2 \rfloor - 1$, compute $\sigma(C_{i_l} \cap N(C_{j_l}))$ and $\sigma(C_{j_l} \cap N(C_{i_l}))$ and denote by $D_l$ the set having the smallest value of $\sigma$. Then, the union $D = \bigcup_l D_l$ satisfies conditions (1) and (2).

THEOREM. *FINDSET executes at most $O(\log n)$ actions. If $n'$ and $m'$, respectively, are the number of vertices and edges of the graph $G'$ induced on the set of decolored vertices, then*

$$n' + m' \leqq (\tfrac{1}{2} + o(1))(n+m).$$

*Proof.* If action $\langle 1 \rangle$ is applied, then the total number of edges and vertices deleted is at least $(n+m)/\log n$; thus, these actions are executed at most $\log n$ times. Obviously, the number of times actions of the second type are applied is also at most $\lceil \log n \rceil$.

Let $D^i$ be the set of vertices that are decolored by the $i$th application of an action of type 2 and let $A = \bigcup D^i$. Then,

$$n' + m' \leqq \sigma(A) \leqq \sum_i \sigma(D^i),$$

$$\sigma(D^i) \leqq \frac{n+m}{2 \log n} \frac{r_i}{\chi(r_i)},$$

and

$$\sigma(A) \leqq \sum_{i=0}^{\lceil \log n \rceil} \frac{n+m}{2 \log n} \frac{r_i}{\chi(r_i)} \leqq \frac{n+m}{2 \log n} \sum_{i=0}^{\lceil \log n \rceil} \frac{r_i}{\chi(r_i)},$$

where $r_i$ is the number of color classes just before the $i$th application of the type 2 action.

To evaluate $\sum_{i=0}^{\lceil \log n \rceil} (r_i/\chi(r_i))$, note that $\chi_i < r_i$ for even $r_i$'s only, and for such $r_i$'s, every application of an action of type 2 reduces the number of color classes by half; if $r_i \geqq 3$ is odd, $r_{i+1} \leqq (r_i/2) + 1$. Thus,

$$\sum_{i=0}^{\lceil \log n \rceil} \frac{r_i}{\chi(r_i)} \leqq \sum_{i=1}^{\lceil \log n \rceil} \frac{2^i}{2^i - 1} \leqq \lceil \log n \rceil + 2,$$

which implies the theorem.

COROLLARY. *Any application of FINDSET yields an independent set $I$, such that*

$$|I \cup N(I)| \geqq (\tfrac{1}{2} - o(1))(n+m).$$

In the next section, we will show that *FINDSET* can be implemented to run in $O(\log^2 n)$ time. This will imply that the running time of the algorithm is $O(\log^3 n)$.

## 4. Implementation of the algorithm.

**4. Implementation of the algorithm.** We will first see how to implement *FINDSET* to run in $O(\log^2 n)$ time on $n + m$ processors. Obviously, for every application of *FINDSET*, action of either type is executed at most $O(\log n)$ times. Thus, we should show that the execution of each action requires only $O(\log n)$ time.

A graph $G$ is represented by a list $L = L(G)$ of its vertices and edges. Each edge is in this list twice, once in each direction. Thus, the length of $L$ is $n + 2m$. For each entry of $L$, there is a processor attached to it. The processors are numbered by $1, \cdots, n +$

$2m$; the first $n$ of them represent the vertices. In addition to its endpoints, each edge knows the colors (if any) of the endpoints and the location of itself given in the other direction. Almost all of the operations are done by sorting this list based on some key. From [AKS], [C] we know that it is possible to sort $m$ records in $O(\log m)$ time on $m$ processors.

To delete a color class $C$, each edge checks the colors of its endpoints to see if one has the same color as $C$. If one does, the edge deletes itself. Then *FINDSET* sorts the list of edges to remove the deleted edges.

To decide whether to do action $\langle 1 \rangle$ or action $\langle 2 \rangle$, *FINDSET* needs to calculate $\sigma(C_i)$, for each $C_i$ in $\phi$. It can calculate the degree of each vertex by sorting the list of edges by their first vertex. It can then sort the list of vertices by color and add up the degrees of the vertices.

When *HALF* performs action $\langle 2 \rangle$, it finds the regular partition $\pi$ that minimizes $\sigma(\pi)$. For this purpose, it needs to know the degree of each vertex. It can calculate them itself, or it can inherit them from the previous step. To calculate $\sigma(\pi)$, *HALF* calculates, for each edge with two colored endpoints, $index(i, j; r)$, where $i$ and $j$ are the colors of the endpoints. It then sorts the edge list by index, breaking ties by first vertex. The different first vertices that occur with a given index $q$ are the vertices in $D^q$ for the partition $\pi_q$. *HALF* then sums the degrees of these vertices to calculate the weights $\sigma(\pi_t)$ $(t = 0, \cdots, \chi(r) - 1)$ and selects the partition with the minimum weight.

Let $q$ be the index of the selected partition $\pi$. At this stage, *HALF* is considering the list $F$ of edges whose index is $q$. For every pair $(C_{i_l}, C_{j_l})$ of $\pi$, the vertices of one of the color classes will be either decolored or recolored when $C_{i_l}$ and $C_{j_l}$ are merged; we call this color class *eligible*. To find eligible classes, *HALF* computes $\sigma_{i_l} = \sigma(C_{i_l} \cap N(C_{j_l}))$ for every $l = 0, \cdots, \lfloor r/2 \rfloor - 1$, and sets class $C_{i_l}$ eligible if and only if $\sigma_{i_j} \leqq \sigma_{j_l}$. Once *HALF* has identified the eligible color classes, it looks at each edge and decolors those vertices $v$ belonging to the eligible color class $C_t$ that are connected to a vertex with color $rev(t, q; r)$. Finally, the pairs of color classes determined by $q$ are merged into single classes by recoloring, for every pair, all the vertices in the color class with the larger color, i.e., each vertex $v$ with original color $c(v)$ changes its color to $rev(c(v), q; r)$ if and only if $rev(c(v), q; r) \leqq c(v)$.

In general, the new color classes do not necessarily have consecutive numbers. To fix this, the vertices are sorted by color, so that the set of colors in use can be determined. Next, *HALF* renumbers the colors and gives each vertex its new color. To update the colors of the first vertex stored with the edges, the list of edges is sorted by first vertex. Then, the pointers to the other representation of the edges are followed to update the color of the second vertex.

Each execution of the main loop of *FINDSET* requires between one and three sorts and $O(\log n)$ time spent doing other miscellaneous work. Thus, *FINDSET* requires only $O(\log^2 n)$ time in all. The number of times *FINDSET* is called is $O(\log n)$ implying that the running time of the algorithm is $O(\log^3 n)$.

So far, we have assumed that $m + n$ processors are available. However, using the processor reduction technique of Miller and Reif [MR], the algorithm can be executed on $(m + n)/\log n$ processors without increasing the running time by more than a constant. For an arbitrary $k > 1$, if there are only $(m + n)/k$ processors, then each real processor can simulate $k$ virtual processors in the algorithm. Since a call to *FINDSET* halves the value $n + m$ of the graph, the number of virtual processors that every real processor simulates decreases by a factor of 2 after each call to *FINDSET*. Thus, there is a constant $C > 0$, such that for every $i = 1, 2, \cdots$, the $i$th call of *FINDSET* is executed in $\leqq C2^{-i} \log^3 n$ time. This increases the running time of the algorithm by a factor of 2.

There is no problem allocating virtual processors to real processors. Each virtual processor is responsible for a single item in list $L$ of vertices and edges in the graph. It is only necessary to reallocate virtual processors after each call to *FINDSET*. The new representation of $G - (I \cup N(I))$ can easily be calculated by sorting. The reallocation can be done even after each execution of the loop in *FINDSET*. While this will speed up the algorithm for a typical graph, the worst-case graphs do not get smaller fast enough to improve the asymptotic performance of the algorithm.

Note that the same technique can be applied to reduce by a factor of $\log n$ the number of processors used by the probabilistic algorithms from [L1] and [ABI].

**5. Conclusion.** An important property of a parallel algorithm is the total work $W$ it does. Obviously, $W \leqq P \times T$, where $P$ is the number of the processors used by the algorithm and $T$ is its running time. Thus, our deterministic algorithm for MIS does $O((m + n)\log^2 n)$ work which is a factor of $\log^2 n$ more than the work done by the obvious sequential algorithm. It would be nice to find an algorithm that does less work. One approach would be to improve the sorting algorithm that our algorithm uses. This might be possible since the keys of all the sorts are small integers. In fact, Reif has an algorithm that sorts $n$ small integers while doing only $O(n)$ work on a randomized concurrent-read, concurrent-write, PRAM [R]. However, if randomization is introduced in the model, then the algorithms from [L1] and [ABI] are preferable. Thus, real improvement would be achieved if better deterministic sorting algorithms were used. Obviously, they would be of interest for other reasons as well. Another approach is to find a way to sort less often. Both approaches appear to be very difficult.

A more promising way to improve the algorithm would be to reduce its running time without increasing the work that it does very much (if at all). It may be possible to reduce the running time of the algorithm by executing different calls to *FINDSET* in parallel. There are several ways to do that; the difficulty seems to be in developing a better analytical technique for estimating the running time.

The dual representation of the edge set may also be useful for other problems, edge-coloring and vertex-coloring being among the first candidates. Another potentially fruitful application of this representation is the problem of constructing an independent set of a *guaranteed* size. In [G], an algorithm was described which runs in $O(\log^3 n)$ time on $O(n + m)$ processors and constructs an independent set with $\geqq n^2/32m$ ($m \geqq n/2$) vertices. Conceivably, an appropriate change in the definition of the *weight* of a set will convert our algorithm into one which builds an independent set containing $\geqq n^2/(2m + n)$ vertices. This is guaranteed by Túran's theorem [T] which also states that this bound is best possible in terms of $n$ and $m$.

**Acknowledgments.** We gratefully thank the three referees for their helpful comments on the first version of the paper.

## REFERENCES

[AKS] M. Ajtai, J. Komlos, and E. Szemeredi, *An $O(n \log n)$ sorting network*, Combinatorica, 3 (1983), pp. 1–19.

[ABI] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. of Algorithms, 7 (1986), pp. 567–583.

[BM] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, North Holland, New York, 1980.

[C] R. Cole, *Parallel merge sort*, Proc. 27th Annual Symp. on Foundation of Computer Science, 1986, pp. 511–516.

[G] M. GOLDBERG, *Parallel algorithms for three graph problems*, CONGRESSUS NUMERANTIUM, 54 (1986), pp. 111–121.

[GS] M. GOLDBERG AND T. SPENCER, *A new parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 18 (1989), pp. 419–427.

[KR] R. M. KARP AND V. RAMACHANDRAN, *Handbook of Theoretical Computer Science*, Preprint, University of California, Berkeley, (1987).

[KW] R. M. KARP AND A. WIGDERSON, *A fast parallel algorithm for the maximal independent set problem*, Proc. 16th ACM Symp. on Theory of Computing, 1984, pp. 266–272.

[L1] M. LUBY, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053.

[L2] ———, *Removing randomness in parallel computation without a processor penalty*, Proc. 29th Annual Symp. on Foundation of Computer Science, 1988, pp. 162–173.

[MR] G. L. MILLER AND J. H. REIF, *Parallel tree contraction and its applications*, Proc. 26th Annual Symp. on Foundation of Computer Science, 1985, pp. 478–489.

[P] N. PIPPENGER, *On simultaneous resource bounds*, Proc. 20th Annual Symp. on Foundation of Computer Science, 1979, pp. 307–311.

[R] J. H. REIF, *An optimal parallel algorithm for integer sorting*, Proc. 26th Annual Symp. on Foundation of Computer Science, (1985), pp. 496–504.

[T] P. TÚRAN, *On the theory of graphs*, Colloq. Math. 3 (1954), pp. 19–30.

[V] U. VISHKIN, *Synchronous parallel computation—a survey*, Preprint, Courant Institute, New York University, New York, (1983).

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 329-343, August 1989

# SINGLE-SUIT TWO-PERSON CARD PLAY III. THE MISÈRE GAME*

J. KAHN†, J. C. LAGARIAS‡, AND H. S. WITSENHAUSEN‡

**Abstract.** Misère End Play is a two-person constant sum game with perfect information. The game uses a deck of cards that consists of a single totally ordered suit of $2n$ cards. To begin play, the deck is divided into two hands of $n$ cards each, held by players Left and Right, with one player designated as having the lead. The player on lead plays one of his cards, and the other player, after seeing it, plays one of his cards. The player with the higher card wins a "trick" and obtains the lead. The cards in the trick are removed and play continues until all cards are played. In Misère End Play each player's goal is to take as few tricks as possible. Misère End Play is shown to have a simple optimal strategy.

**Key words.** combinatorial games, two-person games, card games, computational complexity

**AMS(MOS) subject classifications.** 90D05, 90D42, 68Q25

**1. Introduction.** In previous papers we studied a two-person constant sum perfect information game called the *End Play Game*. The two players are called $L$ (left) and $R$ (right). In the starting position each player holds $n$ cards (his "hand"), which together comprise a set of $2n$ cards (the "deck"). The cards are totally ordered, i.e., the deck of cards has one suit. Both players know the content of both hands and the order. One of the players is designated as having the initial *lead*.

Play proceeds as follows. The leader selects one of his cards, and the other player, *after* he has seen this selected card, plays one of his cards in response. The highest of the two cards scores a "trick" for the player who played it and the lead passes to this player. The two cards played are removed, leaving a new position with hands of $n - 1$ cards and a leader, from which play proceeds until the cards are exhausted. Each player's goal is to win as many tricks as possible.

The game was described in 1929 by Lasker [4], the mathematician and world chess champion, who called it *whistette*. Lasker studied this game as an illustration of simple aspects of play in games of the whist type. He gave rules of play sufficient for hands with a small number of cards, and observed that finding an optimal strategy for hands having a large number of cards might be difficult.

The appeal and apparent subtlety of the End Play Game arises from the presence of the lead, and the rules concerning its movement between the players during the game. In previous papers ([2], [3]) we derived basic properties of the End Play Game and gave evidence suggesting that it indeed may be difficult to compute an optimal strategy for a large number of cards.

In this paper we consider the Misère variant of the game, in which the rules of play are the same, but now each player's objective is to take as few tricks as possible. We call this game the *Misère End Play Game*, or *Misère End Play* for short.

The main result of this paper is that Misère End Play has a simple optimal strategy.

THEOREM 1.1. *For the Misère End Play Game, the following strategy is optimal.*

(1) *When on lead, always lead one's lowest card.*

(2) *When playing to a lead, one should play the highest card in one's hand which is lower than the card led, if this is possible. Otherwise, one should win the trick, playing one's highest card.*

* Received by the editors March 16, 1988; accepted for publication December 8, 1988.

† Rutgers University, New Brunswick, New Jersey. This research was supported in part by grants from the National Science Foundation and a Sloan Research Fellowship.

‡ AT&T Bell Laboratories, Murray Hill, New Jersey.

We will prove this result by induction on the number of cards in each hand. One might expect that this result is simple to establish, but surprisingly this does not seem to be so. The induction we use requires rather complicated hypotheses. The problem we encountered in proving this result is that while the optimal strategy itself is easy to describe, we could not find a simple closed form for the value function $V_n^\epsilon(A, B)$. This prevents us from giving a simple inductive proof using exact formulae for the payoff matrix. Instead, we use a complicated set of regularities of the payoff matrices as inductive hypotheses in order to reach the desired conclusion. We have found examples of payoff matrices which indicate that the induction hypotheses have to be rather complicated (see § 2).

In the End Play Game, having the lead is always a disadvantage that can cost one player at most one trick over what he would take with the same hand when not having the lead ([2, Thm. 3.1]). In Misère End Play, having the lead is also always a disadvantage. However it can potentially cost a player $[\frac{n}{2}]$ tricks in an $n$-trick game. The worst case turns out to be two hands with interlaced cards: $A = \{2n, 2n-2, \cdots, 2\}$, $B = \{2n-1, 2n-3, \cdots, 1\}$. Assuming that Theorem 1.1 holds, it is easy to check that $A$ takes all $n$ tricks if he is on lead, $B$ ducking every trick, while if $B$ is on lead they alternate taking tricks, with $A$ taking $n - [\frac{n}{2}]$ tricks. (We omit a proof that this is in fact the worst case.)

Another variant of the End Play Game arises when each player's objective remains to capture as many tricks as possible, but the rule concerning the lead is changed to require that the player who does *not* capture a trick must lead to the next trick. We call this game the *Reverse-Lead End Play Game*. The Reverse-Lead End Play Game is the same game as Misère End Play, using the following equivalence. One may convert a set of Misère End Play hands to an equivalent set of Reverse-Lead End Play Game by reversing the ordering of the cards. One can then check that an admissible sequence of plays in the Misère Game corresponds to an admissible sequence in the Reverse-Lead game, with the lead behaving properly. In this case a player *not* taking a trick in the Misère game corresponds to a player taking that trick in the equivalent Reverse-Lead game, so that the number of tricks taken by a player in the Reverse-Lead Game equals the number of tricks taken by his opponent in the corresponding Misère Game. Using this equivalence and Theorem 1.1, we find that an optimal strategy for the Reverse-Lead End Play Game is as follows:

(1) When on lead, lead one's highest card.

(2) When playing to a lead, if one can capture a trick, do so with the lowest card that will capture the trick. If one cannot capture the trick, play one's lowest card.

The optimal strategy given in Theorem 1.1 does not extend to Misère End Play with more than one suit. In the multisuited game, the player not on lead is required to play a card in the same suit as the card led ("follow suit"), but if he has no card in this suit, he may play any card ("discard"). The trick is won by the player who played the highest card in the suit led, i.e., play is as in "no-trump" in bridge. The possibility of discards certainly makes Misère End Play with several suits a more complex game. In fact, Misère End Play is a more complicated game even in the special case in which each player holds the same number $n$ of cards in each suit, so that discards cannot occur. Extra complexity arises because not having the lead can be a very valuable commodity. Consider a two-suited game in which the payoff in suit 2 differs by $[\frac{n}{2}]$-tricks, depending on who has the lead, and in which, in suit 1, the effect of the lead on the payoff difference is small. Then the player holding the lowest card in suit 1 may do better than the strategy of Theorem 1.1 by hoarding the lowest card in suit 1 and playing on suit 1 at every opportunity (sacrificing some tricks because he hoards his lowest card) and finally throwing the other player in the lead at the last trick in suit 1 with the hoarded lowest card. The other player might as well play on suit 1 initially also, because he can play on suit 2 only while having the lead, and this is the situation he is trying to avoid. Thus for the Misère Game with more than one suit, even when players have the same number of cards in each suit, it is not clear whether or not there is a simple optimal strategy.

The rest of the paper is organized as follows. In § 2 we study the payoff matrix of the game and prove two preliminary results. One is that having the lead is never advantageous in the game, all other things being equal. The second result (Monotonicity Theorem) is that it is never advantageous for a player to trade a card in one's hand for a higher ranked card. The proof of Theorem 1.1 follows in § 3.

**2. Payoff matrices and the Monotonicity Theorem.** We first establish notation. Let $(A, B)$ denote a pair of hands. The *standard deck* is $\{1, 2, \cdots, 2n\}$ where the cards have the usual ordering on the integers, so that $2n$ is the highest card. A *state* of the game is $(\varepsilon, A, B)$ where $A$ denotes the cards in Left's hand, $B$ denotes the cards in Right's hand, and $\varepsilon = 0$ or 1 indicates who has the lead, where $\varepsilon = 1$ means Left leads. The integer $V_n^\varepsilon(A, B)$ denotes the *value of the state* $(\varepsilon, A, B)$ *to Left*, which is the number of tricks Left has to take with optimal play by both players, and $n = |A| = |B|$ indicates the number of cards each player has in that state.

We may recursively compute the value function as follows. We associate to a pair of hands $(A, B)$ a *payoff matrix* $G(A, B)$, which is an $n \times n$ matrix (where $n = |A| = |B|$) whose rows are indexed by Left's cards in $A$ in decreasing order, and whose columns are indexed by Right's cards in $B$ in decreasing order, and whose $(\mathbf{a}, \mathbf{b})$th entry $G_{\mathbf{a},\mathbf{b}}$ gives the value of the game to Left if Left plays $\mathbf{a}$ and Right plays $\mathbf{b}$ to the first trick. This is

$$G_{\mathbf{a},\mathbf{b}} = [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A-\{\mathbf{a}\}, B-\{\mathbf{b}\}), \tag{2.1}$$

in which $[\mathbf{a}, \mathbf{b}]$ is the value of the first trick, which is

$$[\mathbf{a},\mathbf{b}] = \begin{cases} 1 & \text{if } \mathbf{a} > \mathbf{b}, \\ 0 & \text{if } \mathbf{a} < \mathbf{b}, \end{cases} \tag{2.2}$$

and $[\mathbf{a}, \mathbf{b}]$ also specifies the lead determined by play to the first trick. As an example, Fig. 1 gives the payoff matrix for the hands $(A, B)$ where $A = \{12, 10, 8, 6, 4, 2\}$ and $B = \{11, 9, 7, 5, 3, 1\}$. The line inside the payoff matrix indicates the *capture boundary*, which is the line where the lead changes. Below and to the left of this line $[\mathbf{a}, \mathbf{b}] = 0$.

| **R** | **11** | **9** | **7** | **5** | **3** | **1** |
|---|---|---|---|---|---|---|
| **L** | | | | | | |
| **12** | 6 | 5 | 5 | 4 | 4 | 3 |
| **10** | 3 | 6 | 5 | 4 | 4 | 3 |
| **8** | 3 | 3 | 6 | 5 | 4 | 3 |
| **6** | 3 | 3 | 3 | 6 | 5 | 4 |
| **4** | 4 | 4 | 3 | 3 | 6 | 5 |
| **2** | 5 | 4 | 4 | 3 | 3 | 6 |

FIG. 1. *Payoff matrix giving number of tricks L takes.*

The object of Misère End Play is for each player to minimize the number of tricks he takes. Recursions for the value function can be read off from the payoff matrix. If $R$ is on lead, he wishes to minimize his value, which is equivalent to maximizing Left's payoff, since the total number of tricks in the game is constant. Thus Right wishes to pick the column in the payoff matrix whose smallest entry is largest, so that

$$V_n^0(A,B) = \max_{\mathbf{b}} \min_{\mathbf{a}} [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A-\{\mathbf{a}\}, B-\{\mathbf{b}\}). \tag{2.3}$$

If $L$ is on lead, he picks the row in the payoff matrix whose largest entry is smallest, so that

$$V_n^1(A,B) = \min_{\mathbf{a}} \max_{\mathbf{b}} [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A-\{\mathbf{a}\}, B-\{\mathbf{b}\}). \tag{2.4}$$

The *minimax inequality* asserts that for any rectangular matrix $[m_{ij}]$ one has

$$\max_j \min_i m_{ij} \leqq \min_i \max_j m_{ij}. \tag{2.5}$$

An immediate consequence is that for all $\{A, B\}$ one has

$$V_n^0(A,B) \leqq V_n^1(A,B), \tag{2.6}$$

which asserts that *it can only be advantageous for Left not to have the lead if the hands are held fixed*.

The Misère End Play Game has a symmetry in that the game appears the same to Left and Right. More precisely, the game is symmetric under interchanging Left's and Right's hands while also interchanging the lead. Since the value function measures the value in terms of tricks to Left, this symmetry is expressed in terms of the value function as

$$V_n^\varepsilon(A,B) + V_n^{1-\varepsilon}(B,A) = n, \tag{2.7}$$

for $\varepsilon = 0$ or 1.

We introduce some additional terminology. We say $\mathbf{x}$, $\mathbf{y}$ are *consecutive cards* for hands $(A, B)$ if they are adjacent in the total ordering of the deck $A \cup B$. A *block* for hand $A$ (respectively, $B$) in hands $(A, B)$ is a maximal set of consecutive cards in $A$ (respectively, in $B$). For example, for a standard deck $\{1, 2, \cdots, 2n\}$ a block for $A$ has the form $\{\mathbf{x}, \mathbf{x}+1, \cdots, \mathbf{x}+k\} \subseteq A$ where either $\mathbf{x}-1$ is in $B$ or $\mathbf{x} = 1$ and either $\mathbf{x}+k+1$ is in $B$ or $\mathbf{x}+k = 2n$. Two sets of hands $(A, B)$ and $(A', B')$ are *isomorphic* if they have the same number of cards and the identical partial order, i.e., they differ only in the labelling of the cards and not in the ordering of the cards, e.g., $(\{1, 2\}, \{3, 4\})$ is equivalent to $(\{-3, 7/2\}, \{4, 6\})$. Two cards $\mathbf{x}_1$, $\mathbf{x}_2$ are called *equivalent* for $(A, B)$ if they are both in the same block for one of the hands. Equivalent cards are indistinguishable in the play of the hand, i.e., given hands $(A, B)$ with $\mathbf{x}_1$, $\mathbf{x}_2$ being two equivalent cards in $A$ and $\mathbf{y}_1$, $\mathbf{y}_2$ being two equivalent cards in $B$ then the plays of $(\mathbf{x}_1, \mathbf{y}_1)$ and $(\mathbf{x}_2, \mathbf{y}_2)$ to the first trick are the same in the sense that $[\mathbf{x}_1, \mathbf{y}_1] = [\mathbf{x}_2, \mathbf{y}_2]$ and the resulting pairs of $(n-1)$-card hands

$$(A - \{\mathbf{x}_1\}, B - \{\mathbf{y}_1\}) \quad \text{and} \quad (A - \{\mathbf{x}_2\}, B - \{\mathbf{y}_2\})$$

are isomorphic.

Now we justify the intuitively obvious assertion that exchanging a weaker card in Left's hand for a stronger card in Right's hand is always disadvantageous to Left. We first prove this in the case of an exchange of consecutive cards.

THEOREM 2.1. *Given hands* $(A, B)$, *suppose that* $\mathbf{x}, \mathbf{y}$ *are two consecutive cards in* $A \cup B$ *with* $\mathbf{x} < \mathbf{y}$ *and* $\mathbf{x} \in A$ *and* $\mathbf{y} \in B$. *Let* $A' = A - \{\mathbf{x}\} + \{\mathbf{y}\}$ *and* $\mathbf{B}' = B - \{\mathbf{y}\} + \{\mathbf{x}\}$ *be the hands obtained by interchanging* $\mathbf{x}$ *and* $\mathbf{y}$. *Then*

$$(2.8) \qquad V_n^\varepsilon(A',B') \geqq V_n^\varepsilon(A,B); \qquad \varepsilon = 0 \text{ or } 1.$$

*Proof*. We prove the result by induction on $n$. It is clearly true for $n = 1$, since the value is independent of the lead in that situation. Suppose it is true for $n - 1$. Look at the payoff matrices $G = G(A, B)$ and $G' = G(A', B')$. We claim that each entry $G'_{\mathbf{a},\mathbf{b}}$ in $G(A', B')$ is greater than or equal to the corresponding entry $G_{\mathbf{a},\mathbf{b}}$ of $G(A, B)$. Here the row labelled $\mathbf{y}$ of $G$ is identified with the row labelled $\mathbf{x}$ of $G'$ and the column labelled $\mathbf{x}$ of $G$ is identified with the column labelled $\mathbf{y}$ of $G'$. The truth of this claim immediately implies the desired inequality (2.8) via the recursive formulae (2.3) and (2.4) for the value in terms of the payoff matrices.

To prove the claim we consider four cases. Suppose $\mathbf{a} \in A$, $\mathbf{b} \in B$ are two cards unequal to $\mathbf{x}$ and $\mathbf{y}$, respectively. Then

$$\begin{aligned} G'_{\mathbf{a},\mathbf{b}} &= [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A' - \{\mathbf{a}\}, B' - \{\mathbf{b}\}) \\ &\geqq [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A - \{\mathbf{a}\}, B - \{\mathbf{b}\}) = G_{\mathbf{a},\mathbf{b}} \end{aligned}$$

by the inductive hypothesis, since

$$\begin{aligned} (A - \{\mathbf{a}\})' &= A' - \{\mathbf{a}\} = A - \{\mathbf{a}\} - \{\mathbf{x}\} + \{\mathbf{y}\}, \\ (B - \{\mathbf{b}\})' &= B' - \{\mathbf{b}\} = B - \{\mathbf{b}\} - \{\mathbf{y}\} + \{\mathbf{x}\}. \end{aligned}$$

Second, suppose $\mathbf{x} \in A$, $\mathbf{b} \in B$ with $\mathbf{b} \neq \mathbf{y}$. Then

$$\begin{aligned} G'_{\mathbf{y},\mathbf{b}} &= [\mathbf{y},\mathbf{b}] + V_{n-1}^{[\mathbf{y},\mathbf{b}]}(A' - \{\mathbf{y}\}, B' - \{\mathbf{b}\}) \\ &= [\mathbf{x},\mathbf{b}] + V_{n-1}^{[\mathbf{x},\mathbf{b}]}(A - \{\mathbf{x}\}, B - \{\mathbf{b}\}) = G_{\mathbf{x},\mathbf{b}}, \end{aligned}$$

using the facts that $[\mathbf{y}, \mathbf{b}] = [\mathbf{x}, \mathbf{b}]$ because $\mathbf{y}$ and $\mathbf{x}$ are consecutive cards, and once $\mathbf{x}$ is played the hands $(A' - \{\mathbf{y}\}, B' - \{\mathbf{b}\})$ and $(A - \{\mathbf{x}\}, B - \{\mathbf{b}\})$ have the identical ordering of cards. Third, if $\mathbf{a} \in A$, $\mathbf{y} \in B$ and $\mathbf{a} \neq \mathbf{x}$, then

$$G'_{\mathbf{a},\mathbf{y}} = G_{\mathbf{a},\mathbf{x}}$$

by a similar argument. Finally for $\mathbf{x} \in A$, $\mathbf{y} \in B$

$$\begin{aligned} G'_{\mathbf{y},\mathbf{x}} &= 1 + V_{n-1}^1(A,B) \\ &\geqq V_{n-1}^0(A,B) = G_{\mathbf{x},\mathbf{y}}, \end{aligned}$$

using (2.6). This proves the claim. □

We define a *swap* for $(A, B)$ to be the exchange of a card $\mathbf{a} \in A$ with $\mathbf{b} \in B$ with $\mathbf{a} < \mathbf{b}$, leading to new hands $(A', B')$ with $A' = A - \{\mathbf{a}\} + \{\mathbf{b}\}$, $B' = B - \{\mathbf{b}\} + \{\mathbf{a}\}$. Theorem 2.1 proved that a swap of consecutive cards cannot decrease Left's value. Now define $(A, B)$ to be *superior* to $(A', B')$, written $(A, B) \geqq_s (A', B')$, if $(A, B)$ can be obtained from $(A', B')$ by a series of swaps of consecutive cards. Theorem 2.1 immediately implies that

$$(2.9) \qquad (A,B) \geqq_s (A',B') \Rightarrow V_n^\varepsilon(A',B') \geqq V_n^\varepsilon(A,B); \qquad \varepsilon = 0 \text{ or } 1.$$

It is relatively easy to prove the following result.

THEOREM 2.2 (Monotonicity Theorem). *Given hands* $(A, B)$ *and suppose that* $\mathbf{a} \in A$ *and* $\mathbf{b} \in B$ *with* $\mathbf{a} < \mathbf{b}$. *Then*

$$V_n^\varepsilon(A-\{\mathbf{a}\}+\{\mathbf{b}\},B-\{\mathbf{b}\}+\{\mathbf{a}\}) \geqq V_n^\varepsilon(A,B); \qquad \varepsilon=0 \text{ or } 1. \tag{2.10}$$

*Suppose that* $\mathbf{c}$ *is a new card not in* $A \cup B$ *and that* $\mathbf{c} > \mathbf{a}$. *Then*

$$V_n^\varepsilon(A-\{\mathbf{a}\}+\{\mathbf{c}\},B) \geqq V_n^\varepsilon(A,B); \qquad \varepsilon=0 \text{ or } 1. \tag{2.11}$$

*Proof.* It suffices to show that

$$(A-\{\mathbf{a}\}+\{\mathbf{b}\},B-\{\mathbf{b}\}+\{\mathbf{a}\}) \quad \text{and} \quad (A-\{\mathbf{a}\}+\{\mathbf{c}\},B)$$

are both superior to $(A, B)$. It is a relatively easy matter to find a series of swaps of consecutive cards that show these, cf. [2, § 4]. □

The superiority relation is a partial ordering. It can be shown fairly easily that for hands $(A, B)$ and $(A', B')$ drawn from a standard deck of $2n$ cards if we define the counting functions

$$\Psi_A(m) = \#\{a \in A : a \geqq m\},$$

then $(A, B)$ is *superior* to $(A', B')$ if and only if the condition

$$\Psi_A(m) \geqq \Psi_{A'}(m)$$

holds for $1 \leqq m \leqq 2n$. We omit the proof of this fact as we make no further use of it.

The Monotonicity Theorem gives the following information about an optimal strategy.

THEOREM 2.3. *Misère End Play has an optimal strategy in which the player following to a lead always does one of the following*:

(i) *Takes the trick with the highest card playable.*

(ii) *Does not take the trick, playing the highest card that will not take the trick.*

This theorem asserts nothing about how the player chooses among alternatives (i) and (ii).

*Proof.* Without loss of generality assume that the hands are $A = \{\mathbf{a}_i\}$ and $B = \{\mathbf{b}_j\}$ with cards ordered in *decreasing* order in both hands, and that Right leads, say card $\mathbf{b}_j$. By Theorem 2.2 the function $V_{n-1}^\varepsilon(A - \{\mathbf{a}_i\}, B - \{\mathbf{b}_j\})$ is an increasing function as $\mathbf{a}_i$ decreases. Hence if Left wins the trick, his value is

$$\min_{\{i: [\mathbf{a}_i,\mathbf{b}_j]=1\}} (1+V_{n-1}^1(A-\{\mathbf{a}_i\},B-\{\mathbf{b}_j\})) = 1+V_{n-1}^1(A-\{\mathbf{a}_1\},B-\{\mathbf{b}_j\}),$$

i.e., Left may as well play his highest card. Similarly, if Left loses the trick, his value is

$$\min_{\{i: [\mathbf{a}_i,\mathbf{b}_j]=0\}} (V_{n-1}^0(A-\{\mathbf{a}_i\},B-\{\mathbf{b}_j\})) = V_{n-1}^0(A-\{\mathbf{a}_k\},B-\{\mathbf{b}_j\})$$

where $\mathbf{a}_k$ is the highest card with $\mathbf{a}_k < \mathbf{b}_j$. Hence Left might as well play $\mathbf{a}_k$. □

We conclude this section with an example of a payoff matrix showing that the value to Left is not a monotone decreasing function of the strength of the card Left leads. This example was computed using a recursive calculation of the value function. The hands are $\{A, B\}$ with $A = \{11, 10, 6, 4, 3, 2\}$ and $B = \{12, 9, 8, 7, 5, 1\}$ and the payoff matrix is given in Fig. 2. It is better for Left to lead 10 or 2 than to lead 6.

While we have no general expression for the value as function of the position, we note in passing the (easy) solution for the case where each player has only two blocks. Without loss of generality assume that Left has the lowest $\alpha$ cards as his bottom block,

| R | 12 | 9 | 8 | 7 | 5 | 1 |
|---|---|---|---|---|---|---|
| L | | | | | | |
| 11 | 1 | 3 | 3 | 3 | 3 | 2 |
| 10 | 1 | 3 | 3 | 3 | 3 | 2 |
| 6 | 2 | 2 | 2 | 2 | 4 | 3 |
| 4 | 2 | 2 | 2 | 2 | 2 | 3 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 |
| 2 | 2 | 2 | 2 | 2 | 2 | 3 |

FIG. 2. *Payoff matrix.*

while Right's lower block consists of the next $\beta$ cards. Thus we are talking about the value

$$V_n^\varepsilon(\{1,\cdots,\alpha\}\cup\{\alpha+\beta+1,\cdots,n+\beta\},\{\alpha+1,\cdots,\alpha+\beta\}\cup\{n+\beta+1,\cdots,2n\}).$$

In the boundary case where any of the four blocks is empty, the value is $(\beta-\alpha)^+$, regardless of the lead. In the "interior" case, where $1<\alpha<n$ and $1<\beta<n$, the value is $(\beta-\alpha+\varepsilon)^+$. Here $(x)^+$ denotes max $(0, x)$.

**3. Optimal strategy for Misère End Play.** In this section we prove Theorem 1.1 by induction on $n$. We make the following inductive hypotheses, in which the hands $(A, B)$ have $n$ cards each, unless otherwise noted.

$(A_n)$ It is optimal, when playing second to a trick, to play one's highest card that will not win the trick, and, if this is impossible, to play one's highest card.

$(B_n)$ It is optimal, when on lead, to play one's lowest card.

$(I_n)$ Suppose Right has the lead with hands $(A, B)$. Let $\mathbf{x}\in A$ be at least as high as the highest card in Left's hand that is lower than Right's highest card, and let $\mathbf{l}$ be a card lower than any card in $A\cup B$. Giving Left the lead and replacing Left's hand with $A-\{\mathbf{x}\}+\{\mathbf{l}\}$ is never bad for Left, i.e.,

(3.1) $$V_n^1(A-\{\mathbf{x}\}+\{\mathbf{l}\},B)\leqq V_n^0(A,B).$$

$(J_n)$ Let $|A|=|B|=n-1$ and let $\mathbf{x}$, $\mathbf{y}$ with $\mathbf{x}<\mathbf{y}$ be two new cards such that $\mathbf{x}$ and $\mathbf{y}$ are consecutive cards in the deck $A\cup B\cup\{\mathbf{x},\mathbf{y}\}$. Then for $\varepsilon=0$ or 1 one has

(3.2) $$\begin{aligned}V_{n-1}^\varepsilon(A,B)+1&\geqq V_n^\varepsilon(A\cup\{\mathbf{y}\},B\cup\{\mathbf{x}\})\\&\geqq V_n^\varepsilon(A\cup\{\mathbf{x}\},B\cup\{\mathbf{y}\})\geqq V_{n-1}^\varepsilon(A,B).\end{aligned}$$

$(K_n)$ Suppose that Right has the lead with hands $(A, B)$. Let $\mathbf{b}$ be the lowest card in $B$ and $\mathbf{t}$ a new card higher than all cards in $A\cup B$. If $B^*=B-\{\mathbf{b}\}+\{\mathbf{t}\}$, then replacing Right's hand with $B^*$ and giving Left the lead can benefit Left by at most one trick, i.e.,

(3.3) $$V_n^1(A,B^*)+1\geqq V_n^0(A,B).$$

The hypotheses $(A_n)$, $(B_n)$ specify the optimal strategy, and the hypotheses $(I_n)$–$(K_n)$ are auxiliary hypotheses necessary to complete the induction.

The hypotheses are easily verified to be true in the base cases $n = 1$ and $n = 2$.

We will complete the induction step by verifying a series of implications. To describe these implications, we use the notations $(A_{\leqq n}) = \bigcup_{i=1}^{n}(A_i)$, $(A_{<n}) = \bigcup_{i=1}^{n-1}(A_i)$ and similar abbreviations to denote certain sets of induction hypotheses. The implications are, in order:

(1) $(K_{n-1}) \Rightarrow (A_n)$.

(2) $(A_{\leqq n}), (B_{<n}), (I_{<n}), (J_{<n}), (K_{<n}) \Rightarrow (B_n)$.

(3) $(A_{\leqq n}), (B_{\leqq n}) \Rightarrow (I_n)$.

(4) $(A_{\leqq n}), (B_{\leqq n}), (I_{n-1}), (J_{n-1}) \Rightarrow (J_n)$.

(5) $(A_{\leqq n}), (B_{\leqq n}), (I_n), (J_{n-1}) \Rightarrow (K_n)$.

A brief inspection shows that these implications together complete the induction step. We will prove these implications as a series of claims.

CLAIM 1. $(K_{n-1}) \Rightarrow (A_n)$.

*Proof.* Without loss of generality suppose that the hands are $(A, B)$ with $A = \{\mathbf{a}_i\}$, $B = \{\mathbf{b}_i\}$ arranged in decreasing order and that Left is on lead and led $\mathbf{a}$. By Theorem 2.3, Right either ducks, playing his highest card $\mathbf{b}$ that does not win the trick, or else wins the trick playing his highest card $\{\mathbf{b}_1\}$. If Right ducks, the payoff to Left is $1 + V_{n-1}^1(A - \{\mathbf{a}\}, B - \{\mathbf{b}\})$. If Right wins, Left's payoff is $V_{n-1}^0(A - \{\mathbf{a}\}, B - \{\mathbf{b}_1\})$. But by the Monotonicity Theorem (applied twice) and $(K_{n-1})$ we have

$$\begin{aligned} 1+V_{n-1}^1(A-\{\mathbf{a}\},B-\{\mathbf{b}\}) &\geqq 1+V_{n-1}^1(A-\{\mathbf{a}\},B-\{\mathbf{b}\}-\{\mathbf{b}_1\}+\{\mathbf{t}\}) \\ &\geqq 1+V_{n-1}^1(A-\{\mathbf{a}\},B-\{\mathbf{b}_n\}-\{\mathbf{b}_1\}+\{\mathbf{t}\}) \\ &\geqq V_{n-1}^0(A-\{\mathbf{a}\},B-\{\mathbf{b}_1\}), \end{aligned} \tag{3.4}$$

where $\mathbf{t}$ is a card higher than all cards in $A \cup B$. So Right can always maximize Left's payoff by ducking with his highest card that does not win the trick, if this option is available. □

CLAIM 2. $(A_{\leqq n}), (B_{<n}), (I_{<n}), (J_{<n}), (K_{<n}) \Rightarrow (B_n)$.

*Proof.* Without loss of generality suppose that Right is on lead. Let $\mathbf{b}$ denote the lowest card in $R$'s hand. Suppose that Right leads a card $\mathbf{d}$. We must show that, under optimal play by both sides, Right can do at least as well leading $\mathbf{b}$ as leading $\mathbf{d}$.

We can analyze the play using the observation that once Right leads, Left's reply may be selected using the already proved induction hypothesis $(A_n)$, and all subsequent optimal plays for Left and Right can be made using the induction hypotheses $(A_{<n}), (B_{<n})$.

The proof is divided into several cases. To describe these, we first note that we may suppose $\mathbf{d}$ is not in $R$'s lowest block, because otherwise the play of $\mathbf{b}$ and $\mathbf{d}$ are equivalent. Then $L$ must have a card lower than $\mathbf{d}$ and we let $\mathbf{c}$ denote $L$'s highest such card. We also let $\mathbf{p}$ denote Left's highest card (which may coincide with $\mathbf{c}$) and $\mathbf{a}$ $L$'s lowest card (which may coincide with, or be equivalent to, $\mathbf{c}$). We denote by $\mathbf{q}$ Right's highest card (which may coincide with, or be equivalent to, $\mathbf{d}$). We also suppose without loss of generality that $\mathbf{d}$ is the lowest card in its block, noting that $\mathbf{c}$ is the highest in its block, by definition.

*Case* 1. $L$ has the lowest card.

Then $\mathbf{a} < \mathbf{b} < \mathbf{c} < \mathbf{d}$.

| | | | | | | |
|---|---|---|---|---|---|---|
| $L$ | $\cdots$ | | $\mathbf{c}$ $\cdots$ | | $\cdots$ | $\mathbf{a}$ |
| $R$ | $\cdots$ | $\mathbf{d}$ | | $\cdots$ $\mathbf{b}$ | | |

By $(A_n)$ if $R$ leads **d**, then $L$ ducks with **c**; then by $(B_{n-1})$, $(A_{n-1})$ on the next trick $R$ leads **b**, and $L$ ducks with **a**. If $R$ instead leads **b** at trick 1, then $L$ ducks with **a**. At trick 2, $R$ has the *option* to play **d**, whence, by $(B_{n-1})$, $L$ will duck with **c**, and this leads to exactly the same position and payoff after 2 tricks. And $R$ may have a better lead at trick 2 than this. So $R$ does at least as well leading his lowest card **b** at trick 1.

*Case* 2. $R$ has the lowest card.

This divides into 2 subcases according to whether **d** is equivalent to Right's highest card **q** or not.

*Case* 2.a. **d** is equivalent to $R$'s highest card **q**.

Thus $R$ is leading from his top block. This divides into two subcases, depending on whether **c** is Left's highest card or not.

*Case* 2.a.1. **c** is Left's highest card (**c** = **p**).

| | |
|---|---|
| $L$ | c ··· |
| $R$ | ··· d ··· b |

If $R$ leads **d**, $L$ ducks with **c**, with payoff $V^0_{n-1}(A - \{\mathbf{c}\}, B - \{\mathbf{d}\})$. If $R$ leads **b**, then $L$ wins with **c**, with payoff $1 + V^1_{n-1}(A - \{\mathbf{c}\}, B - \{\mathbf{b}\})$. Now if **t** denotes a card higher than all cards in $A \cup B$, then by $(K_{n-1})$ we have

$$\begin{aligned}V^0_{n-1}(A-\{\mathbf{c}\},B-\{\mathbf{d}\}) &\leqq 1+V^1_{n-1}(A-\{\mathbf{c}\},B-\{\mathbf{d}\}-\{\mathbf{b}\}+\{\mathbf{t}\})\\ &= 1+V^1_{n-1}(A-\{\mathbf{c}\},B-\{\mathbf{b}\}),\end{aligned}$$

where the equality follows by the equivalence of **d** and **t**. Hence $R$ does at least as well to lead **b**, which completes Case 2.a.1.

*Case* 2.a.2. **c** is lower than Left's highest card **p**.

This divides into two subcases, according to whether **c** is or is not equal to **a**.

*Case* 2.a.2.a. Left has two or more cards lower than **d**, so **a** is not equal to **c**.

This divides into two subcases according to whether the number of cards below **a** is 1 or more.

*Case* 2.a.2.a.1. **b** is the only card below **a**.

| | |
|---|---|
| $L$ | p ··· c ··· a |
| $R$ | ··· d ··· b |

If $R$ leads **d**, $L$ ducks with **c**. Then at trick 2 $R$ leads **b**, $L$ wins with **p**. If instead $R$ leads **b** at trick 1, $L$ wins with **p**, and, at trick 2, $L$ leads **a** which $R$ wins with **d**. The payoffs are $1 + V^1_{n-2}(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{d}\})$ and

$$1+V^0_{n-2}(A-\{\mathbf{a},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\}),$$

respectively. Now

$$V^1_{n-2}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\})\leqq V^0_{n-2}(A-\{\mathbf{a},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\})$$

holds by $(I_{n-2})$, which applies since if $B' = B - \{\mathbf{b}, \mathbf{d}\}$ then **a** is the lowest card in the deck $(A - \{\mathbf{c}, \mathbf{p}\}) \cup B'$ and **c** is at least as high as the highest card in Left's hand in $(A - \{\mathbf{a}, \mathbf{p}\}, B')$ lower than the highest card in $B'$. Hence $R$ does at least as well to lead **b** as to lead **d**.

*Case* 2.a.2.a.2. Right has at least two cards lower than Left's lowest card **a**, of these cards **b** is the lowest and **b**′ the highest.

| | |
|---|---|
| $L$ | p ··· c ··· a |
| $R$ | ··· d ··· b′ ··· b |

If $R$ leads $\mathbf{d}$, $L$ ducks with $\mathbf{c}$; at trick 2 $R$ leads $\mathbf{b}$, and $L$ wins with $\mathbf{p}$; the payoff is $1 + V^1_{n-2}(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{d}, \mathbf{b}\})$. If $R$ leads $\mathbf{b}$, then $L$ wins with $\mathbf{p}$; at trick 2 $L$ leads $\mathbf{a}$, $R$ ducks with $\mathbf{b}'$. The payoff to $L$ is then $2 + V^1_{n-2}(A - \{\mathbf{a}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{b}'\})$. To see that

$$V^1_{n-2}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{d},\mathbf{b}\}) \leqq 1 + V^1_{n-2}(A-\{\mathbf{a},\mathbf{p}\},B-\{\mathbf{b},\mathbf{b}'\}), \tag{3.5}$$

holds, we observe that by $(J_{n-1})$ we have

$$V^1_{n-2}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{d},\mathbf{b}\}) \leqq V^1_{n-1}(A-\{\mathbf{p}\},B-\{\mathbf{b}\}),$$

and

$$V^1_{n-1}(A-\{\mathbf{p}\},B-\{\mathbf{b}\}) \leqq 1 + V^1_{n-2}(A-\{\mathbf{a},\mathbf{p}\},B-\{\mathbf{b},\mathbf{b}'\})$$

since $\mathbf{a}$ and $\mathbf{b}'$ are consecutive cards with $\mathbf{a} > \mathbf{b}'$. Combining these two inequalities proves (3.5) which completes Case 2.a.2.a.2. and thereby Case 2.a.2.a.

*Case* 2.a.2.b. Left has exactly one card lower than $\mathbf{d}$, so $\mathbf{a}$ and $\mathbf{c}$ are equal.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $L$ | p | ··· | | | c | | |
| $R$ | | q | ··· | d | | ··· | b |

Now all of Left's cards but $\mathbf{c}$ are higher than *all* of Right's cards. If Right leads $\mathbf{d}$, then $L$ takes $n - 1$ tricks in all. But $L$ always takes at least $n - 1$ tricks with any lead, since $L$ has $n - 1$ cards higher than all cards in $R$'s hand. Hence $R$ may as well lead $\mathbf{b}$, which completes Case 2.a.2.b. This settles Cases 2.a.2. and 2.a.

*Case* 2.b. $\mathbf{d}$ is not equivalent to $R$'s highest card $\mathbf{q}$.

Thus Right leads from one of his inner blocks (neither the highest, nor the lowest.) This again divides into two cases depending on whether $\mathbf{a}$ equals $\mathbf{c}$ or not.

*Case* 2.b.1. Left has two or more cards lower than $\mathbf{d}$, so $\mathbf{a}$ is lower than $\mathbf{c}$.

This divides as earlier according to the size of $\mathbf{b}$'s block.

*Case* 2.b.1.a. Right has at least two cards lower than Left's lowest card $\mathbf{a}$, of which $\mathbf{b}'$ is the highest.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $L$ | p | ··· | ··· | | c | ··· | a | | | | |
| $R$ | q | ··· | | ··· | | d | | ··· | b′ | ··· | b |

If $R$ leads $\mathbf{d}$, $L$ ducks with $\mathbf{c}$; at trick 2 $R$ leads $\mathbf{b}$, and $L$ takes with $\mathbf{p}$. The payoff to Left is then $1 + V^1_{n-2}(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{d}, \mathbf{b}\})$. On the other hand, if $R$ leads $\mathbf{b}$, then $L$ wins with $\mathbf{p}$ and leads $\mathbf{a}$, which $R$ ducks with $\mathbf{b}'$, giving the payoff $2 + V^0_{n-2}(A - \{\mathbf{a}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{b}'\})$. Now we note that

$$V^1_{n-2}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\}) \leqq 1 + V^0_{n-\varepsilon}(A-\{\mathbf{a},\mathbf{p}\},B-\{\mathbf{b},\mathbf{b}'\})$$

holds by the same argument as in Case 2.a.2.a.2., using hypothesis $(J_{n-1})$.

*Case* 2.b.1.b. Right has exactly one card $\mathbf{b}$ lower than Left's lowest card $\mathbf{a}$.

| | | | | | | |
|---|---|---|---|---|---|---|
| $L$ | ··· | | c | ··· | a | |
| $R$ | ··· | d | | ··· | | b |

If $R$ leads $\mathbf{d}$, $L$ ducks with $\mathbf{c}$; at trick 2 $R$ leads $\mathbf{b}$, which $L$ wins with $\mathbf{p}$; at trick 3 $L$ leads $\mathbf{a}$, which $R$ wins with $\mathbf{q}$. The payoff is

$$1 + V^0_{n-3}(A-\{\mathbf{a},\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d},\mathbf{q}\}).$$

On the other hand, if $R$ leads **b**, then $L$ wins with **p**; at trick 2 $L$ leads **a**, which $R$ wins with **q**. At trick 3 $R$ may *choose* to lead **d**, in which case $L$ will duck with **c**, giving the payoff

$$1+V_{n-3}^{0}(A-\{\mathbf{a},\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d},\mathbf{q}\})$$

to Left, and $R$ might do better by leading differently at the third trick. Hence $R$ can do at least as well leading **b** as leading **d**. This completes Case 2.b.1.b and thereby Case 2.b.1.

*Case* 2.b.2. Left has exactly one card lower than **d**, i.e., **a** is equal to **c**.

This again divides depending on the number of cards less than **a**.

*Case* 2.b.2.a. $R$ has at least two cards lower than Left's lower card **a** (=**c**), of which **b**′ is the highest.

| | | | | |
|---|---|---|---|---|
| $L$ | **p** ··· ··· | **c** | | |
| $R$ | **q** ··· | ··· **d** | **b**′ | ··· **b** |

If Right leads **d**, $L$ ducks with **c**; at trick 2 $R$ leads **b**, which $L$ wins with **p**. The payoff to Left is

$$1+V_{n-2}^{1}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\}).$$

If Right leads **b**, $L$ wins with **p**; at trick 2 $L$ leads **c**, and $R$ ducks with **b**′. The payoff to Left is then

$$2+V_{n-2}^{1}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{b}'\}).$$

Since **c** is the only card in Left's hand between **d** and **b**, the hands $(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{d}\})$ and $(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{b}'\})$ are equivalent. Hence Right prefers to lead **b**. This completes Case 2.b.2.a.

*Case* 2.b.2.b. Right has exactly one card lower than Left's lowest card **a** (= **c**).

| | | | |
|---|---|---|---|
| $L$ | **p** ··· ··· | **c** | |
| $R$ | **q** ··· | ··· **d** | **b** |

If $R$ leads **d**, $L$ ducks with **c**; at trick 2 $R$ leads **b**, and $L$ wins with **p**. Left's payoff is

$$1+V_{n-2}^{1}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\}).$$

If $R$ leads **b**, $L$ wins with **p**; at trick 2 $L$ leads **c**, and $R$ wins with **q**. Left's payoff is then

$$1+V_{n-2}^{0}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{q}\}).$$

It now suffices to show that

$$(3.6) \qquad V_{n-2}^{0}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{q}\}) \geqq V_{n-2}^{1}(A-\{\mathbf{c},\mathbf{p}\},B-\{\mathbf{b},\mathbf{d}\}).$$

To obtain this, we apply $(I_{n-2})$ with Right and Left's hands interchanged and with $\mathbf{x} = \mathbf{q}$ to obtain

$$V_{n-2}^{1}(B-\{\mathbf{b},\mathbf{d},\mathbf{q}\}+\{\mathbf{l}\},A-\{\mathbf{c},\mathbf{p}\}) \leqq V_{n-2}^{0}(B-\{\mathbf{b},\mathbf{d}\},A-\{\mathbf{c},\mathbf{p}\}).$$

Observe that the hands $(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{q}\})$ and $(A - \{\mathbf{c}, \mathbf{p}\}, B - \{\mathbf{b}, \mathbf{d}, \mathbf{q}\} + \{\mathbf{l}\})$ are isomorphic, so that the last inequality implies

$$V_{n-2}^{1}(B-\{\mathbf{b},\mathbf{q}\},A-\{\mathbf{c},\mathbf{p}\}) \leqq V_{n-2}^{0}(B-\{\mathbf{b},\mathbf{d}\},A-\{\mathbf{c},\mathbf{p}\}).$$

Substituting (2.7) into both sides of this inequality yields (3.6). This completes Case 2.b.2.b., which completes the proof of Cases 2.b.2., 2.b., and 2., thus proving Claim 2. □

CLAIM 3. $(A_{\leqq n}), (B_{\leqq n}) \Rightarrow (I_n)$.

*Proof.* We are given hands $(A, B)$ and a card $\mathbf{x} \in A$ which is at least as high as the highest card in $A$ which is lower than the highest card in $B$, and a new card $\mathbf{1}$ not in $A \cup B$ which is lower than all cards in $A \cup B$. We wish to prove

$$V_n^1(A-\{\mathbf{x}\}+\{\mathbf{1}\},B) \leqq V_n^0(A,B). \tag{3.7}$$

By the Monotonicity Theorem it suffices to prove this inequality in the case that $\mathbf{x}$ is the highest card in $A$ that is lower than the highest card $\mathbf{q}$ in $B$. Call $(0, A, B)$ the *Before* game and $(1, A - \{\mathbf{a}\} + \{\mathbf{1}\}, B)$ the *After* game. Using $(A_n)$, $(B_n)$ in the After game, Left leads $\mathbf{1}$, Right wins with $\mathbf{q}$. Hence

$$V_n^1(A-\{\mathbf{x}\}+\{\mathbf{1}\},B) = V_{n-1}^0(A-\{\mathbf{x}\},B-\{\mathbf{q}\}). \tag{3.8}$$

In the Before game, Right may choose to lead $\mathbf{q}$ (which is generally suboptimal) in which case Left will duck playing $\mathbf{x}$. Left's value is at least as low as it would be if Right played optimally, hence

$$V_n^0(A,B) \geqq V_{n-1}^0(A-\{\mathbf{x}\},B-\{\mathbf{q}\}). \tag{3.9}$$

Then (3.8) and (3.9) imply the desired bound (3.7). □

CLAIM 4. $(A_{\leqq n}), (B_{\leqq n}), (I_{n-1}), (J_{n-1}) \Rightarrow (J_n)$.

*Proof.* The central inequality in $(J_n)$ holds by monotonicity and the third inequality follows from the first via (2.7). Thus we need only prove the first inequality. We are given hands $(A, B)$ with $|A| = |B| = n - 1$ and $\mathbf{x}$, $\mathbf{y}$ two new cards such that $\mathbf{x}$, $\mathbf{y}$ are consecutive cards in the deck $A \cup B \cup \{\mathbf{x}, \mathbf{y}\}$. By renumbering cards if necessary, we suppose that $A \cup B \cup \{\mathbf{x}, \mathbf{y}\}$ is a standard deck $\{1, 2, \cdots, 2n\}$, in which case $\mathbf{y} = \mathbf{x} + 1$. We are to show

$$V_{n-1}^{\varepsilon}(A,B)+1 \geqq V_n^{\varepsilon}(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\}). \tag{3.10}$$

We call the game with $(A, B)$ the *Original game* and the game with hands

$$(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\})$$

the *Augmented game*. We consider several cases, as follows.

*Case* 1. Left is on lead ($\varepsilon = 1$).

In the Augmented game Left may choose to lead $\mathbf{x} + 1$, which may be a suboptimal play, in which case by $(A_n)$ Right ducks playing $\mathbf{x}$. The payoff to Left playing this way, with optimal play thereafter, is $1 + V_{n-1}^1(A, B)$, hence

$$V_n^1(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\}) \leqq 1 + V_{n-1}^1(A,B).$$

*Case* 2. Right is on lead ($\varepsilon = 0$).

*Case* 2a. Right's lowest card $\mathbf{b}$ is lower than $\mathbf{x}$.

In both the original game and augmented game Right leads $\mathbf{b}$ by $(B_n)$. In the original game Left plays a card $\mathbf{a} \in A$, and in the augmented game Left may choose to play $\mathbf{a} \in A$, though he might have a better play. Hence

$$V_n^0(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\}) \leqq [\mathbf{a},\mathbf{b}] + V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A+\{\mathbf{x}+1\}-\{\mathbf{a}\},B+\{\mathbf{x}\}-\{\mathbf{b}\})$$

$$V_{n-1}^0(A,B) = [\mathbf{a},\mathbf{b}] + V_{n-2}^{[\mathbf{a},\mathbf{b}]}(A-\{\mathbf{a}\},B-\{\mathbf{b}\}).$$

By the induction hypothesis $(J_{n-1})$, we have

$$V_{n-1}^{[\mathbf{a},\mathbf{b}]}(A+\{\mathbf{x}+1\}-\{\mathbf{a}\},B+\{\mathbf{x}\}-\{\mathbf{b}\})\leqq 1+V_{n-2}^{[\mathbf{a},\mathbf{b}]}(A-\{\mathbf{a}\},B-\{\mathbf{b}\}).$$

These last three inequalities imply that

$$V_n^0(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\})\leqq 1+V_{n-1}^0(A,B)$$

holds in this case.

*Case* 2b. $\mathbf{x}$ is lower than all cards in Right's hand and Left holds a card lower than $\mathbf{x}$.

Then Left's highest card lower than $\mathbf{x}$ is necessarily $\mathbf{x} - 1$. In the Augmented game, by $(A_n)$, $(B_n)$, Right leads $\mathbf{x}$, $L$ ducks playing $\mathbf{x} - 1$. Hence

$$V_n^0(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\})=V_{n-1}^0(A+\{\mathbf{x}+1\}-\{\mathbf{x}-1\},B).$$

But with $\mathbf{x}$ gone, $\mathbf{x} + 1$ and $\mathbf{x} - 1$ are equivalent cards, so $(A + \{\mathbf{x} + 1\} - \{\mathbf{x} - 1\}, B)$ is isomorphic to $(A, B)$, so that

$$V_n^0(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\})=V_{n-1}^0(A,B)$$

in this case, so that (3.10) holds.

*Case* 2c. $\mathbf{x}$ is lower than all cards in $A \cup B$.

Hence $\mathbf{x} = 1$. Let $\mathbf{p}$ be Left's highest card in $A$. In the Augmented game by $(A_n)$, $(B_n)$ we have: Right leads $\mathbf{x}$, Left wins with $\mathbf{p}$. Hence

$$V_n^0(A+\{\mathbf{x}+1\},B+\{\mathbf{x}\})=1+V_{n-1}^1(A+\{\mathbf{x}+1\}-\{\mathbf{p}\},B).$$

Now

$$V_{n-1}^1(A+\{\mathbf{x}+1\}-\{\mathbf{p}\},B)\leqq V_{n-1}^0(A,B)$$

holds by $(I_{n-1})$, because $\mathbf{p}$ is at least as high as the highest card in $A$ less than the highest card in $B$, and $\mathbf{x} + 1 = 2$ is lower than all cards in $A \cup B = \{3, 4, \cdots, 2n\}$. Substituting this inequality in the previous equation yields (3.10) in this case.

These cases are exhaustive and prove Claim 4. □

CLAIM 5. $(A_{\leqq n})$, $(B_{\leqq n})$, $(I_n)$, $(J_{n-1}) \Rightarrow (K_n)$.

*Proof.* We are given hands $(A, B)$ with $\mathbf{b}$ being the lowest card in $B$ and $\mathbf{t}$ a new card higher than all cards in $A \cup B$. We must show that

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})+1\geqq V_n^0(A,B). \tag{3.11}$$

We call the state $(0, A, B)$ the *Before game* and $(1, A, B - \{\mathbf{b}\} + \{\mathbf{t}\})$ the *After game*.

*Case* 1. Left has a card lower than $\mathbf{b}$.

Let $\mathbf{a}'$ be Left's highest card lower than $\mathbf{b}$, and $\mathbf{a}$ Left's lowest card. They are both in Left's lowest block by hypothesis, so they are equivalent cards. (They may in fact be equal.)

Before, with Right leading:

| | | | | | |
|---|---|---|---|---|---|
| $L$ | $\cdots$ | | $\mathbf{a}'$ | $\cdots$ | $\mathbf{a}$ |
| $R$ | $\cdots$ | $\mathbf{b}$ | | | |

After, with Left leading:

| | | | | | |
|---|---|---|---|---|---|
| $L$ | | $\cdots$ | $\mathbf{a}'$ | $\cdots$ | $\mathbf{a}$ |
| $R$ | $\mathbf{t}$ | $\cdots$ | | | |

In the Before game, by hypotheses $(A_n)$ and $(B_n)$, Right leads **b**, and Left ducks playing **a**$'$. Hence

$$V_n^0(A,B)=V_{n-1}^0(A-\{\mathbf{a}'\},B-\{\mathbf{b}\}). \tag{3.12}$$

In the After game, Left leads **a**, and Right wins with **t**. Hence

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})=V_{n-1}^0(A-\{\mathbf{a}\},B-\{\mathbf{b}\}). \tag{3.13}$$

Since **a** and **a**$'$ are equivalent cards, $(A-\{\mathbf{a}'\}, B-\{\mathbf{b}\})$ and $(A-\{\mathbf{a}\}, B-\{\mathbf{b}\})$ are isomorphic games, so the right sides of (3.12) and (3.13) are equal, whence

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})=V_n^0(A,B).$$

*Case* 2. **b** is the lowest card in $A \cup B$.

This divides into two cases depending on the size of Right's lowest block.

*Case* 2a. **b** is the only card lower than Left's lowest card **a**.

Before, with Right leading:

| | | | | | |
|---|---|---|---|---|---|
| $L$ | | **p** | $\cdots$ | **a** | |
| $R$ | $\cdots$ | | $\cdots$ | | **b** |

After, with Left leading:

| | | | | | |
|---|---|---|---|---|---|
| $L$ | | | **p** | $\cdots$ | **a** |
| $R$ | **t** | $\cdots$ | | $\cdots$ | |

Let $B' = B - \{\mathbf{b}\}$. In the Before game, Right leads **b**, which Left wins with his highest card **p**. Thus

$$V_n^0(A,B)=1+V_{n-1}^1(A-\{\mathbf{p}\},B'). \tag{3.14}$$

In the After game Left leads **a**, which Right wins with **t**. Hence

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})=V_{n-1}^0(A-\{\mathbf{a}\},B'). \tag{3.15}$$

Now

$$V_{n-1}^1(A-\{\mathbf{p}\},B')\leqq V_{n-1}^0(A-\{\mathbf{a}\},B') \tag{3.16}$$

holds by $(I_{n-1})$ on taking $A-\{\mathbf{a}\}$ as $A$, $B'$ as $B$ and $(\mathbf{p},\mathbf{a})$ as $(\mathbf{x},\mathbf{l})$ in $(I_{n-1})$. Combining (3.14)-(3.16) yields

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})+1\geqq V_n^0(A,B)$$

for this case.

*Case* 2b. Right has at least two cards lower than Left's lowest card **a**. Of these cards, **b**$'$ is the highest.

Before, with Right leading:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $L$ | | **p** | $\cdots$ | **a** | | | |
| $R$ | $\cdots$ | | $\cdots$ | | **b**$'$ | $\cdots$ | **b** |

In the After game, with Left leading:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $L$ | | | **p** | $\cdots$ | **a** | | |
| $R$ | **t** | $\cdots$ | | $\cdots$ | | **b**$'$ | $\cdots$ |

In the Before game Right leads $\mathbf{b}$, which Left wins with his highest card $\mathbf{p}$. At trick 2, Left leads $\mathbf{a}$, and Right ducks by playing $\mathbf{b}'$. Hence

$$V_n^0(A,B)=2+V_{n-2}^1(A',B'),$$

where $A'=A-\{\mathbf{a},\mathbf{p}\}$, $B'=B-\{\mathbf{b},\mathbf{b}'\}$.

In the After game, Left leads $\mathbf{a}$, and Right ducks with $\mathbf{b}'$ at trick 1, hence

$$V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})=1+V_{n-1}^1(A'+\{\mathbf{p}\},B'+\{\mathbf{t}\}).$$

Thus to show that

$$1+V_n^1(A,B-\{\mathbf{b}\}+\{\mathbf{t}\})\geqq V_n^0(A,B)$$

we need only to show that

$$V_{n-1}^1(A'+\{\mathbf{p}\},B'+\{\mathbf{t}\})\geqq V_{n-2}^1(A',B').$$

This follows from $(J_{n-1})$ since $\mathbf{t}$ is equivalent to the lowest card in $B'+\{\mathbf{t}\}$ higher than $\mathbf{p}$. This completes Case 2b.

These cases are exhaustive and prove Claim 5. □

This completes the induction step and the proof of Theorem 1.1. □

**Acknowledgment.** The first author thanks Aeryeung Moon for helpful discussions.

## REFERENCES

[1] E. Berlekamp, J. Conway, and R. Guy, *Winning Ways*, Vols. I and II, Academic Press, New York, 1982.

[2] J. Kahn, J. C. Lagarias, and H. S. Witsenhausen, *Single-suit two-person card play*, Intl. J. Game Theory, 16 (1987), pp. 291–320.

[3] ———, *Single suit two-person card play* II: *Domination*, Order, 5 (1988), pp. 45–60.

[4] E. Lasker, *Encyclopedia of Games*, *Vol*. I., *Card Strategy*, E. P. Dutton & Co., New York, 1929.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 344-349, August 1989

# DIVISORS WITHOUT UNIT-CONGRUENT RATIOS*

D. J. KLEITMAN†

**Abstract.** This paper addresses the question: how large can a collection of divisors of a square free integer be, if any pair $x$ and $y$ in the collection so that $x$ divides $y$, has $y/x$ congruent to 1 mod $p$? It is shown that if $n$ has as many prime factors congruent to $k$ and $1/k$ mod $p$, and an even number congruent to $p - 1$, then the sum of the largest two binomial coefficients on $n$ is an upper bound to the size of the collection. Asymptotic bounds are also obtained.

**Key words.** divisors, finite sets, combinatorics

**AMS(MOS) subject classification.** 05A05

**1. Introduction.** Erdös [1] recently raised the following question: Suppose we have a prime number $p$ and a collection, $C$, of divisors of a square-free natural number $m$, with $m$ a product of $n$ prime factors. Suppose further that, if a member $x$ of $C$ divides another, $y$, then their quotient, $y/x$, is not congruent to 1 mod $p$. How large then can $C$ be? Erdös conjectured that, if the divisors of $m$ are uniformly distributed among the nonzero congruence classes mod $p$, there is an upper bound of the form $cC(n, [n/2])$, for some constant $c$.

If, in this situation, the prime factors of $m$ are all congruent to 1 mod $p$, then no member of $C$ divides another, and the well-known theorem of Sperner [2] implies that the cardinality of $C$ is at most $C(n, [n/2])$. If, on the other hand, all the prime factors of $m$ are congruent to some nonunit $j$ mod $p$, then we can have the sum of the $p - 2$ largest $n$-binomial coefficients members in $C$ by taking all divisors of $m$ that are of the corresponding $p - 2$ largest ranks among divisors of $m$, where the rank of a divisor is the number of its prime factors. For large $n$ this lower bound is asymptotic to $(p - 2)C(n, [n/2])$ and no upper bound $cC(n, [n/2])$ with $c$ independent of $p$ is possible.

In this note, we provide an affirmative answer to this conjecture. We show further that $C(n, [n/2]) + C(n, [n/2] + 1)$ is the exact upper bound here when, for each $k$, the same number of prime divisors of $n$ are congruent to $k$ and to $1/k$ mod $p$ (and none are congruent to 1; the same bound holds but is not best possible when some of the prime factors are congruent to 1 mod $p$).

In the next section the nature of this argument is discussed, and our results are formally stated. Proofs are contained in § 3 which is followed by a discussion of the open questions mentioned above.

**2. Method and results.** Our basic approach involves defining a new partial order in which the requirements on $C$ imply that $C$ must be an antichain. We then use various methods that are known for bounding the size of antichains in order to obtain our bounds.

The $2^n$ divisors of $m$ form a partially ordered set isomorphic to the Boolean algebra of subsets of a set under the order relation of division. We define a new partial order, $P$, on the same elements, with $a > b$ if $b$ divides $a$ and $a = b$ (mod $p$). This order retains some, but not all, of the ordered pairs of the Boolean algebra; there are at least $(p - 1)$

* Received by the editors July 20, 1987; accepted for publication (in revised form) November 23, 1988. This research was supported in part by Air Force Office of Scientific Research contract AFOSR-86-0078 and National Science Foundation contract DMS-86-06225.

† Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

connected components of this order consisting of those divisors in each congruence class mod $p$. It is evident that any $C$ obeying the conditions under consideration here must be an antichain in this order.

At first glance, the order $P$ appears to have a structure that is not easily analyzable. However, by weakening it still further, we obtain an order, $Q$, that we can handle quite easily.

We define $Q$ as follows: $a$ covers $b$ in $Q$ if $a = b \pmod p$ and $a = bcd$ with $c$ and $d$ primes, (so that $cd = 1 \pmod p$). It follows that the elements of $Q$ that are products of an even number of primes, which we call $Q_e$, are entirely disconnected from those, forming $Q_o$, that are products of an odd number of primes.

We shall show that in a wide variety of circumstances, these two orders are "almost" Sperner, in that the largest possible size of an antichain in either is not much more than the largest rank size, where rank of a divisor is the number of its prime factors. Our first main result follows from these arguments.

THEOREM 1. *Suppose that $m$ is the product of $n$ prime factors, that $s_j$ of these are congruent to $j$* mod $p$ *with*

$$s_j = n/(p-2) + x_j \quad \text{for } j > 1,$$

$$s_0 = s_1 = 0,$$

*and for all $j$, $|x_j| < un^{1/2+\delta}$ for some constant $u$ and $\delta < \frac{1}{6}$. Then with no $a$, $b$ in $C$ with $a = bd$, $d = 1$* (mod $p$), *$|C|$ can be at most asymptotic for large $n$ to $2C(n, [n/2])$. In the order in which $a$ covers $b$ if $a = bcd$ with $c$ and $d$ primes and $cd = 1$* mod $p$, *the even and odd ranks each are asymptotically* LYM.

THEOREM 2. *If in Theorem* 1, *we have*

$$s_j = s_{1/j} \quad \text{for } p-2 > j > 1,$$

$$s_{p-1} \quad \text{is even},$$

*and*

$$s_0 = s_1 = 0,$$

*then $C$ can have no more than $C(n, n/2) + C(n, n/2 + 1)$ elements. If $s_1 = q$, then the exact upper bound on $C$ is*:

$$\sum_{j=-n}^{j=n} (C(n-q, 2j+(n-q)/2) + C(n-q, 2j+1+(n-q)/2))C(q, [q/2]+j).$$

THEOREM 3. *The conclusions of Theorem* 1 *hold if $m$ has, in addition to factors as indicated, an arbitrary number of prime factors congruent to* 1 mod $p$.

Alternate proofs of Theorems 1 and 3 may be obtained by applying Theorem 2. In fact, the following strengthening of them immediately follows from Theorem 2.

THEOREM 4. *Suppose that, with the notation of Theorem* 1, *as $n$ increases, the ratio*

$$\sum_{k=2}^{p-2} |s_k - s_{1/k}|/2n$$

*approaches zero. Then the conclusions of Theorems* 1 *and* 3 *follow.*

It follows from Theorems 1 and 3, or 4, that if the prime factors of $m$ are randomly distributed over the congruence classes mod $p$, we have a bound that is asymptotic to $2C(n, [n/2])$ almost always.

In proving these we first notice that under the given circumstances, the degrees of all but a small fraction of the elements of a given rank in the graph of the covering relation for $Q$ are not far from the same. This implies an "asymptotic LYM [3], [4], [5] property" for these partial orders, which allows both the stated conclusion of Theorem 1 and its extension in Theorem 3.

We prove Theorem 2 by exhibiting an explicit partition of the elements of the order into chains with at most the given number of chains.

Theorem 4 is obtained by applying Theorem 2 to a portion of the factors that approaches one, for every combination of the other factors.

The LYM inequality for the Boolean algebra of subsets of a set is the statement that the sum, over sets sizes of the proportion of sets of that size in an antichain, is at most one.

Its original proof is very simple and pretty: consider all maximal chains of subsets. Each can contain, at most, one member of an antichain. Therefore, the proportion, $r$, of these chains that contain a member of an antichain $C$ is the sum, over set sizes, of the proportion of chains containing a member of $C$ of that size. Since all sets of any one size appear in the same number of these chains, the latter sum is the sum over set sizes of the proportion of sets of that size in $C$, while the former cannot exceed one.

$$\begin{aligned} 1 \geqq r &= \sum_i (\text{proportion of chains with } i\text{-set in } C) \\ &= \sum_i (\text{proportion of } i\text{-sets in } C). \end{aligned}$$

Our general approach for the present problems is to mimic this argument for the partial order $Q$. Each element of rank $k$ (with rank defined as in the Boolean algebra on the same elements) will not lie in the same number of chains, but in almost the same number, which will replace the last equality by an "almost" equality.

$$\begin{aligned} 1 \geqq r &= \sum_i (\text{proportion of chains with } i\text{-rank member in } C) \\ &\approx \sum_i (\text{proportion of } i\text{-rank in } C). \end{aligned}$$

This argument can generally be used if one can find any collection of chains such that $C$ can contain only one member of each chain, and "asymptotically all" elements of our order of given rank appear in "asymptotically the same number" of chains.

We shall prove Theorem 2 by first treating the prime factors in each pair $k$ and $1/k$ of congruence classes together showing that, using divisibility order relation, these can be partitioned into chains such that two elements an even distance apart in the same chain are ordered in $Q$. We then observe that all products of the even and odd subchains of these chains, with one term per pair of factors (or single factor for $k = p - 1$) form products of chains which can themselves be partitioned into chains in a standard way. In the resulting chain partition of the entire set of divisors, each chain has a member of rank $[n/2]$ or $[n/2] + 1$ from which the conclusion immediately follows.

## 3. Proofs.

*Proof of Theorem* 1. The divisors of $m$ can be classified by the number of their prime factors in each congruence class mod $p$. A factor for which these numbers are $d_2, \cdots, d_{p-1}$, will have "rank" $r$ given by the sum of these numbers. We assume that $m$ has $n/(p-2)$ prime factors in each of these classes. A proportion

$2^{-n/(p-2)}C(n/(p-2)+x_j, s)$ of the divisors of $n$ will have exactly $s$ prime factors in any one specific congruence class.

Well known properties of binomial coefficients imply that there are a proportionally small number of factors of $m$ for which any of the $d_j$ values differ by more than $2(n/(p-2))^{1/2}\log n$ from $n/2(p-2)+x/2$.

If we write

$$d_j = n/2(p-2)+y_j,$$

we may, in asymptotic considerations, limit ourselves $y_j$ whose magnitudes are all less than $2(n/(p-2))^{(1/2)+\delta}\log n$.

We set $s = n/(p-2)$ in the remainder of this discussion.

The number of divisors that are covered by a divisor having parameters $d_j$ in chains in $Q$ as described in the previous section is:

$$\left(d_{p-1}(d_{p-1}-1)+\sum_{j=2}^{p-2} d_j d_{1/j}\right)\Big/2$$

or

$$\sum_{j=2}^{p-1}(s/2+y_j)(s/2+y_{1/j})/2-d_{p-1}/2,$$

which, for any positive $\varepsilon$, reduces to

$$(p-2)s^2/8+s(r-n/2)/2+o(ps^{1+2\delta+\varepsilon}).$$

It is only the last term here which differs among elements of the same rank, and this term is small by $s^{1-2\delta-\varepsilon}$ compared to the leading term. The number of divisors of $m$ that cover a given divisor of rank $r$ is, by the same arguments:

$$\sum_{j=2}^{p-1}(s/2+x_j-y_j)(s/2+x_{1/j}-y_{1/j})/2-s-x_{-1}+d_{-1}/2,$$

which reduces to

$$(p-2)s^2/8-s(r-n/2)/2+o(ps^{1+2\delta+\varepsilon}).$$

It follows that the number of coverings by (or of) an element of $Q$ of a given rank is, for asymptotically all of the elements at near middle rank, almost constant.

If we count the number of chains extending from rank $n/2-n^{1/2+z}$ to rank $n/2+n^{1/2+z}$ for $0<z<\frac{1}{2}-2\delta-\varepsilon$ that contain a given element, we find that it is within an asymptotically negligible factor the same as that containing any other at the same rank.

Theorem 1 follows by the LYM argument already indicated from these statements.

*Proof of Theorem* 2. The previous argument was based entirely upon the relative uniformity of degrees within one rank in the covering graph here.

We can draw a stronger conclusion by making a more detailed examination of the structure of this order.

Greene and Kleitman [6] showed that if one has a $k$-element set $X$, and colors $[k/2]$ of its elements red, we can partition $X$'s subsets into chains in such a way that the difference between any two sets in the same chain, whose sizes have the same parity, is half red.

We indicate the proof of this lemma here for completeness: The "standard" partition of subsets into chains is obtained by representing the subsets as 0-1 sequences, replacing

0's by left parentheses, 1's by right parentheses, closing all parentheses that close under the standard rule for closing parentheses, and placing subsets in the same chain if and only if they have the identical pattern of closed parentheses in the identical locations in their sequences. If the order of the components used to define the sequences alternates in color, the chains so produced will have the claimed property.

For our purposes, we let the red elements correspond to prime factors congruent to $k$ and the others be prime factors congruent to $1/k$. The conclusion of the Greene-Kleitman Theorem then tells us that two elements of the same chain with the same parity have quotient congruent to 1 mod $p$. (When $k = p - 1$, $1/k = k$, and an arbitrary ordering of the components yields the same result.)

Elements of these chains for fixed $k$ correspond to divisors of $m$ that are products of factors all of which are congruent to $k$ or $1/k$. Divisors of $m$ in general can be corresponded with elements in all possible products of these chains for all the various $k$.

We partition each of our chains into two chains, so that the elements within each all have the same rank parity, and are therefore chains in $Q$. When $s_k = s_{1/k}$ as we assume here (and $s_{p-1}$ is even), all of the resulting chains are symmetric about middle rank.

The products of all the half chains for different $k$ values have the order properties of products of chains, and these can themselves be partitioned into chains in the standard deBruijn, Tengbergen, and Kruyswijk [7] manner. One obtains chains, still symmetric about middle rank, in which successive elements in the same chain are two ranks apart. Each chain contains either an element of rank $n/2$ or one of rank $n/2 + 1$.

The partitions into symmetric chains here give rise to many more conclusions about properties of divisors than those mentioned here.

If there are prime factors congruent to 1 mod $p$, we partition them into chains in the standard way, and take the product of each of these with the chains obtained for the rest of the divisors.

The theorem follows immediately from the obvious properties of the chains involved here.

*Proof of Theorem* 3. If $m$ has prime factors congruent to 1 mod $p$, we can consider its divisors to form a partial order that is the direct product of the Boolean algebra on such divisors, and the orders $Q_e$ and $Q_o$ discussed above, on its other divisors. Harper's [8], [9] Theorem tells us that the product of two orders each obeying the LYM property, and each having rank sizes such that the square of the size of the $k$th rank exceeds the product of the sizes of the $k - 1$st and $k + 1$st, also obeys the LYM property. It is straightforward to extend this theorem to yield the asymptotic LYM property of a product given asymptotic LYM-ness of the factors. The conclusion obtainable in this way is actually stronger than the statement of the theorem. A maximal collection $C$ has an asymptotic size-bound of

$$\sum_{j=-n}^{j=n} (C(n-q,2j+(n-q)/2)+C(n-q,2j+1+(n-q)/2))C(q,[q/2]+j),$$

when there are $q$ prime factors of $n$ congruent to 1 mod $p$.

*Proof of Theorem* 4. Our plan is to apply Theorem 2 not to $m$ but to $m$ divided by enough prime factors for that theorem to apply. For each $k$, we drop $s_k - s_{k-1}$ factors when that quantity is positive (and one for $p - 1$ if $s_{p-1}$ is odd). If $q$ prime factors are dropped all together, the conclusion of Theorem 2 yields a bound of $(C(n - q, (n - q)/2) + C(n - q, (n - q)/2 + 1))2^q$. This is asymptotic to our desired expression when the ratio $q/n$ approaches zero.

Similar arguments apply when $s_1 > 0$.

**4. Discussion.** Our results show that when sums of the number of prime divisors congruent to $k$ and $1/k$ mod $p$ are much larger than their differences, then the size of $C$ is asymptotic to twice the largest binomial coefficient on $n$. Suppose, therefore, we have the opposite situation, in which for each $k$, either $s_k$ or $s_{1/k}$ vanishes, but that the nonzero $s$'s were all equal. We might ask, what bound on the size of $C$ must exist under these circumstances?

A lower bound to the maximum possible size of $C$ of the sum of the largest 3 binomial coefficients on $n$ is obvious under these circumstances. It might be interesting to investigate upper bounds in this case and in its generalizations.

## REFERENCES

[1] P. ERDÖS, *private communication*, Clemson SIAM Meeting, May, 1986.

[2] E. SPERNER, *Ein Satz über Untermengen einer Endlichen Menge*, Math. Z., 27 (1928), pp. 544–548.

[3] D. LUBELL, *A short proof of Sperner's lemma*, J. Combin. Theory Ser., 1 (1966), p. 299.

[4] K. YAMAMOTO, *Logarithmic order of the free distributive lattice*, J. Math. Soc. Japan, 6 (1954), pp. 343–353.

[5] L. D. MESHALKIN, *A generalization of Sperner's theorem on the number of subsets of a finite set*, Theory Prob. Appl., 8 (1963), pp. 203–204.

[6] C. GREENE AND D. J. KLEITMAN, *Strong versions of Sperner's theorem,* J. Combin. Theory Ser., 20 (1976), pp. 80–88.

[7] N. DE BRUIJN, C. TENGBERGEN, AND D. KRUYSWIJK, *On the set of divisors of a number*, Nieuw. Arch. Wisk., 23 (1951), pp. 191–193.

[8] L. HARPER, *private communication*, 1968.

[9] W. N. HSIEH AND D. J. KLEITMAN, *Normalized matching in direct products of partial orders*, Stud. Appl. Math., 52 (1973), pp. 285–289.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 350-365, August 1989

# PERTURBATIONS OF SHIFTS OF FINITE TYPE*

D. A. LIND†

**Abstract.** Shifts of finite type describe the infinite trips on a labeled graph, and provide theoretical models for data storage and transmission. The consequences of forbidding a fixed word to occur, which can be considered as a small change or perturbation in the system, are investigated. This situation arises in prefix synchronized codes, where a certain prefix, used to synchronize code words, is forbidden to occur in the rest of the word. If $T$ is the adjacency matrix of the graph, and $\lambda_T$ is its spectral radius, then forbidding a word of length $k$ results in a drop in spectral radius that lies between two positive constants times $\lambda_T^{-k}$. The zeta function summarizes the number of possible periodic trips. The author gives an explicit calculation of the zeta function for the resulting subshift, which involves the characteristic polynomial of $T$, a cofactor of $tI - T$, and the correlation polynomial of the word. A modification of the Knuth–Morris–Pratt pattern matching algorithm shows that this calculation can be done in time that is linear in the word length, answering a question of Bowen and Lanford. The structure of this correlation polynomial is used to obtain sharp bounds on the degree of the denominator of the zeta function. The Jordan form of the higher order presentations of the shift of finite type is also computed, and that of the perturbation in many cases. Most of the results were discovered experimentally with a computer.

**Key words.** shift of finite type, symbolic dynamics, channel capacity, topological entropy, zeta function, perturbation, correlation polynomial

**AMS(MOS) subject classifications.** primary: 54H20, 58F20, 58F30; secondary: 05A15, 28D20, 34C35

## 1. Introduction.

A *dynamical system* is a pair $(X, \sigma)$, where $X$ is a topological space and $\sigma$ is a continuous map from $X$ to itself. Shifts of finite type are combinatorially defined dynamical systems that arise naturally in the study of data storage and transmission [1]. They also play a central role in the analysis of other dynamical systems such as diffeomorphisms of manifolds [3], an idea which goes back to Hadamard's analysis in 1898 of geodesic flows. Roughly speaking, shifts of finite type are the topological analogues of the probabilistic Markov chains used in information theory.

There are two equivalent definitions of shifts of finite type. The first definition begins with a nonnegative integral matrix $T$, and forms the directed graph $G_T$ having $T_{ij}$ distinct edges from vertex $i$ to vertex $j$. Let $\mathscr{E}$ denote the set of edges of $G_T$. The space $X_T \subset \mathscr{E}^{\mathbb{Z}}$ consists of all bi-infinite sequences from $\mathscr{E}$ that give an allowed trip on $G_T$. The map $\sigma_T$: $X_T \rightarrow X_T$ shifts a sequence one edge to the left. As an example, let $T = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, so that $G_T$ has two vertices, say labeled 0 and 1, and three edges, say labeled $\xi_{00}$, $\xi_{01}$, and $\xi_{10}$. A sequence $\cdots \xi_{i_{-1}j_{-1}} \xi_{i_0 j_0} \xi_{i_1 j_1} \cdots$ is in $X_T$ exactly when, for every $k$, the terminal vertex of $\xi_{i_k j_k}$ agrees with the initial vertex of $\xi_{i_{k+1} j_{k+1}}$, i.e., when $j_k = i_{k+1}$.

The second definition starts with a finite alphabet $A$, and a finite collection $\mathscr{F}$ of "forbidden" finite words over $A$. The space $X_{\mathscr{F}} \subset A^{\mathbb{Z}}$ consists of all bi-infinite sequences from $A$ that do not contain any word from $\mathscr{F}$. The shift map $\sigma_{\mathscr{F}}$: $X_{\mathscr{F}} \rightarrow X_{\mathscr{F}}$ acts as before. For example, if $A = \{0, 1\}$ and $\mathscr{F} = \{11\}$, then $X_{\mathscr{F}}$ is the set of all bi-infinite sequences of 0's and 1's that do not contain consecutive 1's.

In each definition there is a distance function on bi-infinite sequences defined by $d(\{x_k\}, \{y_k\}) = 2^{-n}$, where $n$ is the largest integer such that $x_k = y_k$ for $-n \leqq k \leqq n$. Thus two points are close if their coordinates agree on a large symmetric interval of indices. The shift map is continuous with respect to the topology induced by the metric $d$.

* Received by the editors June 13, 1988; accepted for publication (in revised form) January 20, 1989.

† Department of Mathematics, University of Washington, Seattle, Washington 98195. This work was supported by National Science Foundation grants DMS-8320356 and DMS-8706248.

It is usual to consider two dynamical systems to be the "same" if one can be obtained from the other by a continuous relabeling of points. More precisely, we say that $(X_1, \tau_1)$ is *topologically conjugate* to $(X_2, \tau_2)$ if there is a continuous map $\varphi: X_1 \to X_2$ with a continuous inverse such that $\tau_2\varphi = \varphi\tau_1$. The two examples of shifts of finite type given above are topologically conjugate using the map $\varphi: X_T \to X_{\mathcal{F}}$ given by $\varphi(\{\xi_{i_kj_k}\}) = \{i_k\}$, so that $\varphi$ assigns to a sequence of edges the corresponding sequence of their initial vertices.

A *topological invariant* of dynamical systems is an object (real number, group, etc.) assigned to every dynamical system that is the same for topologically conjugate systems. Examples of such invariants are entropy and the zeta function, which are defined below.

Our two definitions of shifts of finite type are equivalent in the sense that any dynamical system constructed using one definition is topologically conjugate to a dynamical system constructed by the other. If $(X_T, \sigma_T)$ is a system from the first definition, let $\mathcal{E}$ be the alphabet, and let $\mathcal{F}$ be the set of pairs of edges satisfying the condition that the terminal vertex of the first edge is not the initial vertex of the second. Then $(X_{\mathcal{F}}, \sigma_{\mathcal{F}})$ is topologically conjugate to $(X_T, \sigma_T)$. Conversely, if $(X_{\mathcal{F}}, \sigma_{\mathcal{F}})$ is constructed using the second definition, it is possible to determine, as in § 2, a directed graph with adjacency matrix $T$ so that $(X_T, \sigma_T)$ is conjugate to $(X_{\mathcal{F}}, \sigma_{\mathcal{F}})$.

We shall use the first definition of shift of finite type throughout. When a shift arises by forbidding words, we will convert this to a conjugate shift defined by a nonnegative integer matrix using the process described in § 2. We shall also assume throughout that for each pair of nodes there is a path from the first to the second or, equivalently, that $\sigma_T$ is topologically transitive. To avoid trivial exceptions, we also require $T \neq [1]$.

Prefix synchronized codes [7], [8] can be described as follows. Select a fixed block $B$, and choose code words of the form $BA$ subject to the constraint that $B$ occurs in $BAB$ only as the first and last subblock. Thus an occurrence of $B$ guarantees that a decoder is at the beginning of a code word. In estimating the number of code words available, we are quickly led to the study of the shift of finite type obtained by forbidding $B$ to occur. If $B$ is long, this shift can be considered as a small change in the original shift.

Let $B \in \mathcal{E}^k$ be an allowed path in the graph $G_T$ of length $k = |B|$. The shift of finite type obtained from $(X_T, \sigma_T)$ by forbidding $B$ to occur can be presented by a matrix we denote by $T\langle B\rangle$ that is typically much larger than $T$. The details of this construction are given in § 2. For the purposes of this paper we regard passing from $\sigma_T$ to $\sigma_{T\langle B\rangle}$ as a small perturbation of $\sigma_T$, with the perturbation tending to zero as $|B| \to \infty$. Our main theorems give precise information about the resulting changes in topological entropy, zeta function, and Jordan form.

To describe the change in entropy, let $T$ have spectral radius $\lambda_T$. The topological entropy $h(\sigma_T)$ is then $\log \lambda_T$. Standard Perron–Frobenius theory [16, Thm. 1.1(*e*)] shows $h(\sigma_T) - h(\sigma_{T\langle B\rangle}) > 0$. During the investigation [5] of the automorphism group of $\sigma_T$, it was necessary to prove this difference tends to zero as $|B| \to \infty$ [5, Thm. 5.1]. The analysis of the speed of convergence was one motivation for this paper. We show in Theorem 3 that there are computable constants $c_T$, $d_T > 0$ so that for all blocks $B$ with sufficiently large length $|B|$ we have

$$c_T\lambda_T^{-|B|} < h(\sigma_T) - h(\sigma_{T\langle B\rangle}) < d_T\lambda_T^{-|B|}.$$

The zeta function of a map was introduced by Artin and Mazur [2] to conveniently summarize periodic point data. If $N_n$ denotes the number of points in $X_T$ fixed by $\sigma_T^n$, then the zeta function of $\sigma_T$ is defined as

$$\zeta_T(t) = \exp\left(\sum_{n=1}^{\infty} \frac{N_n}{n} t^n\right).$$

Bowen and Lanford [4] showed $\zeta_T(t) = 1/\det(I - tT)$, which we reformulate differently as follows. If $f(t) \in \mathbb{Z}[t, t^{-1}]$ is a Laurent polynomial, choose the unique $d \in \mathbb{Z}$ so that $t^d f(t) \in \mathbb{Z}[t]$ with nonzero constant term. Let $f(t)^\times$ denote this $t^d f(t)$. If $\chi_T(t) = \det(tI - T)$ is the characteristic polynomial of $T$, then clearly

$$\zeta_T(t) = \frac{1}{\chi_T(t^{-1})^\times}.$$

To describe the zeta function of the perturbed matrix, we first recall the notion of correlation polynomial introduced by J. H. Conway and exploited in a series of papers by Guibas and Odlyzko [8]–[10]. If $B \in \mathscr{E}^k$, the *correlation polynomial* of $B$ is $\phi_B(t) = \sum_{j=1}^k c_j t^{j-1}$, where $c_j = 1$ if $B$ overlaps itself in $j$ symbols, and $c_j = 0$ otherwise. A recursive description of the set of possible correlation polynomials is given in [9], having the surprising consequence that this set is independent of $\mathscr{E}$ for $|\mathscr{E}| \geqq 2$. These polynomials arise in a wide range of applications, including the computation of the number of strings of a given length omitting a fixed block [8], the analysis of pattern matching algorithms [9], the study of nerve impulses in crayfish [6], and some astonishing probability calculations concerning "paradoxical" nontransitive games [10].

Suppose $B$ is a path from node $i$ to node $j$. Call $B$ *reduced* if no proper subblock of $B$ determines $B$. Let $\operatorname{cof}_{ij}(tI - T)$ denote $(-1)^{i+j}$ times the determinant of the matrix $tI - T$ with its $i$th row and $j$th column removed. We show in Theorem 1 that

$$\zeta_{T\langle B\rangle}(t) = \frac{1}{[\chi_T(t^{-1})\phi_B(t^{-1}) + \operatorname{cof}_{ij}(t^{-1}I - T)]^\times}. \tag{1.1}$$

When $T = [n]$, so $\sigma_T$ is the full $n$-shift, every block is reduced, and our result simplifies to

$$\zeta_{T\langle B\rangle}(t) = \frac{1}{(1 - nt)\phi_B(t^{-1})t^{|B|-1} + t^{|B|}}.$$

Since $T\langle B\rangle$ has size that is exponential in $|B|$, it is not clear whether there is an algorithm to compute $\zeta_{T\langle B\rangle}(t)$ in time polynomial in $|B|$. But since there is a trivial algorithm to compute $\phi_B(t)$ in time that is quadratic in $|B|$, one consequence of Theorem 1 is the existence of a quadratic-time algorithm to compute $\zeta_{T\langle B\rangle}(t)$. This is sharpened in Corollary 2 to a linear algorithm based on the Knuth–Morris–Pratt pattern matching algorithm [13], answering a question raised by Bowen and Lanford [4, p. 45]. Also, although there are roughly $\lambda_T^k$ blocks of length $k$, Guibas and Odlyzko show that there are only $O(k^{c \log k})$ distinct correlation polynomials for these blocks. Thus another consequence of our calculation, stated in Corollary 3, is that the number of distinct zeta functions produced by omitting blocks of length $k$ is only $O(k^{c \log k})$. In Corollary 4 we show that the number of distinct zeta functions produced by deleting blocks of length $k$ is the same for all full shifts.

Because of possible divisibility by powers of $t$ in $\chi_T(t)$, $\phi_B(t)$, and $\operatorname{cof}_{ij}(tI - T)$, together with possible cancellations, the formula (1.1) does not immediately yield the degree of $\zeta_{T\langle B\rangle}^{-1}(t)$. Using machinery in §§ 2–3, combined with the recursive characterization of $\phi_B(t)$ from [9], we show in Theorem 2 that

$$|B| - 4r \leqq \deg \zeta_{T\langle B\rangle}^{-1} \leqq |B| + r - 1,$$

where $r$ is the size of $T$, and that the upper bound is attained precisely when

$$(\det T)\phi_B(0) + \operatorname{cof}_{ij}(-T) \neq 0. \tag{1.2}$$

In particular, the upper bound obtains for all blocks in a full shift.

The Jordan form of $T$ is not a topological invariant, but our analysis allows the determination in Theorem 4 of the Jordan form of all higher block presentations of $T$. Also, if (1.2) holds, then the Jordan form of $T\langle B\rangle$ is obtained from that of the $k$-block presentation of $T$ by deleting a single Jordan nilpotent block of size $k-1$. Again, every block in a full shift satisfies this condition.

The paper is organized as follows. We begin in § 2 by setting up the machinery for higher block presentations of $T$. In § 3 we determine a low-dimensional subspace containing the "interesting part" of $T\langle B\rangle$ and compute the matrix of $T\langle B\rangle$ with respect to a certain natural basis. This matrix contains the companion matrix of $\phi_B(t)$ as a principal submatrix and is used in § 4 to compute $\zeta_{T\langle B\rangle}(t)$ and to estimate $\deg \zeta^{-1}_{T\langle B\rangle}(t)$. In § 5 we first motivate the estimate on decrease in entropy using an elementary but apparently little-known fact on the rate of change in an eigenvalue with respect to the matrix entries. We then prove the estimate by using the calculation of $\chi_{T\langle B\rangle}(t)$ together with the recursive structure of $\phi_B(t)$. Finally, we apply this machinery in § 6 to derive the results on Jordan forms mentioned above.

Most of our results were discovered experimentally using the Matlab interactive linear algebra computer program. Jim Scherer (private communication) has indicated to us an alternative approach to an upper bound on $\deg \zeta^{-1}_{T\langle B\rangle}$, but his results are not as sharp as Theorem 2. Razmik Karabed (private communication) has described an interesting method for obtaining an exponential upper bound in Theorem 3, but his method does not yield either the best exponent or the lower bound.

**2. Higher order presentations.** In this section we introduce the linear algebra machinery we will use to analyze $T\langle B\rangle$. Our treatment of higher block presentations varies slightly from the standard one, since we use a consistent treatment for all nonnegative integer matrices rather than just 0-1 matrices.

Let $T=[T_{ij}]$ be an $r\times r$ matrix over the nonnegative integers. Let $\mathcal{S}=\{1,\cdots,r\}$ be the indexing set of $T$, and call the elements of $\mathcal{S}$ the *1-states*. For $1\leqq p\leqq T_{ij}$, introduce *1-symbols* $\xi^p_{ij}$, and let $\mathcal{E}=\{\xi^p_{ij}: 1\leqq p\leqq T_{ij}, 1\leqq i,j\leqq r\}$. Thus $\mathcal{E}$ is a labeling of the edges of the directed graph determined by $T$. Define beginning and ending maps $\beta$, $\eta$: $\mathcal{E}\rightarrow\mathcal{S}$ by $\beta(\xi^p_{ij})=i$ and $\eta(\xi^p_{ij})=j$. Thus

$$T_{ij}=|\{\xi\in\mathcal{E}:\beta(\xi)=i \text{ and } \eta(\xi)=j\}|.$$

Put

$$X_T=\{x=(x_n)\in\mathcal{E}^{\mathbb{Z}}:\eta(x_n)=\beta(x_{n+1}) \text{ for } n\in\mathbb{Z}\},$$

and define the shift $\sigma_T$: $X_T\rightarrow X_T$ by $(\sigma_T x)_n=x_{n+1}$. Then $\sigma_T$ is a homeomorphism of the compact totally disconnected metric space $X_T$. The pair $(X_T,\sigma_T)$ is the *shift of finite type* determined by $T$.

Next we recode points in $X_T$ by $k$-blocks of symbols. For $k\geqq 1$ let

$$\mathcal{B}_k=\mathcal{B}_k(X_T)=\{\xi_1\cdots\xi_k\in\mathcal{E}^k:\eta(\xi_m)=\beta(\xi_{m+1}) \text{ for } 1\leqq m\leqq k-1\}$$

be the set of allowed $k$-blocks in $X_T$. By convention we put $\mathcal{B}_0=\mathcal{S}$. For $k\geqq 1$, let $\mathcal{S}^{[k]}=\mathcal{B}_{k-1}$ denote the *k-states* and $\mathcal{E}^{[k]}=\mathcal{B}_k$ be the *k-symbols*. Note that $\mathcal{S}^{[1]}=\mathcal{S}$ and $\mathcal{E}^{[1]}=\mathcal{E}$. For $k\geqq 2$, define $\beta$, $\eta$: $\mathcal{E}^{[k]}\rightarrow\mathcal{S}^{[k]}$ by $\beta(\xi_1\cdots\xi_k)=\xi_1\cdots\xi_{k-1}$ and $\eta(\xi_1\cdots\xi_k)=\xi_2\cdots\xi_k$. Since $\mathcal{S}^{[k]}=\mathcal{E}^{[k-1]}=\mathcal{B}_{k-1}$, we may compose various $\beta$'s and $\eta$'s in any order, and they clearly commute.

We will now define the $k$th *order presentation matrix* $T^{[k]}$ of $T$, which is indexed by $k$-states $\mathcal{S}^{[k]}$. For $A,B\in\mathcal{S}^{[k]}$ define

$$(T^{[k]})_{AB}=|\{C\in\mathcal{E}^{[k]}:\beta(C)=A \text{ and } \eta(C)=B\}|.$$

Note that $T^{[1]}$ is just $T$. If $k \geqq 2$, then $T^{[k]}$ is a 0-1 matrix. Let $\mathbb{R}^{[k]}$ denote $\mathbb{R}^{|\mathscr{S}^{[k]}|}$, with its basis the elementary vectors $\{e_A^{[k]} : A \in \mathscr{S}^{[k]}\}$. Then $T^{[k]}$ is a linear map of $\mathbb{R}^{[k]}$. It is notationally convenient to have all matrices act on the *right*. Thus, for $k \geqq 2$,

$$e_A^{[k]} T^{[k]} = \sum_{\{B \in \mathscr{S}^{[k]} : \beta B = \eta A\}} e_B^{[k]}.$$

PROPOSITION. *The systems* $(X_T, \sigma_T)$ *and* $(X_{T^{[k]}}, \sigma_{T^{[k]}})$ *are topologically conjugate.*

*Proof.* This holds for $k = 1$ by definition. Suppose $k \geqq 2$. Denote $\alpha: X_{T^{[k]}} \to \mathscr{E}^{\mathbb{Z}}$ by $\alpha((C_n)_{n\in\mathbb{Z}}) = (\beta^{k-1}C_n)_{n\in\mathbb{Z}}$. Clearly, $\alpha$ is continuous. Since $\eta C_n = \beta C_{n+1}$, we have

$$\eta(\beta^{k-1}C_n) = \beta^{k-1}\eta C_n = \beta(\beta^{k-1}C_{n+1}),$$

so the image of $\alpha$ is contained in $X_T$. Clearly, $\sigma_T \alpha = \alpha \sigma_{T^{[k]}}$. Define $\gamma: X_T \to X_{T^{[k]}}$ by $(\gamma x)_m = x_m x_{m+1} \cdots x_{m+k-1} \in \mathscr{B}_k$. Then $\gamma$ is continuous, and $\alpha\gamma$ and $\gamma\alpha$ are the identity maps. This shows $\alpha$ is a topological conjugacy of $\sigma_{T^{[k]}}$ with $\sigma_T$. □

**3. Invariant subspaces for perturbations.** Let $B \in \mathscr{B}_k(X_T)$, where we assume from now on that $k \geqq 2$. Forbidding $B$ in $X_T$ is the same as forbidding the transition from $\beta B$ to $\eta B$ in $T^{[k]}$. Thus a matrix presentation of the resulting shift of finite type is $T^{[k]} - E_{\beta E, \eta B}$, where $E = E_{\beta B, \eta B}$ has a 1 in the $(\beta B, \eta B)$th place and is zero elsewhere. Let $T\langle B\rangle = T^{[k]} - E$ denote this perturbed matrix. In this section we will find a $T^{[k]}$-invariant subspace $V_k \subset \mathbb{R}^{[k]}$ on which $T^{[k]}$ mimics $T$. Then we extend $V_k$ to a $T\langle B\rangle$-invariant subspace $W_{k,B}$ containing the "interesting part" of $T\langle B\rangle$. Finally, we compute the matrix of $T\langle B\rangle$ with respect to a certain basis for $W_{k,B}$ and show how the companion matrix for the correlation polynomial of $B$ appears as a principal submatrix.

Define $\psi: \mathbb{R}^{[1]} \to \mathbb{R}^{[k]}$ by

$$\psi(e_i^{[1]}) = \sum_{\{A \in \mathscr{S}^{[k]} : \beta^{k-1}A = i\}} e_A^{[k]}, \qquad 1 \leqq i \leqq r. \tag{3.1}$$

Note that the sum is over $k$-states $A$ whose initial *state* (not initial symbol) is $i$. Let $V_k$ be the image of $\psi$ in $\mathbb{R}^{[k]}$. The *eventual range* of a linear map on a vector space is the intersection of the images of its nonnegative powers.

LEMMA 1. *The map* $\psi$ *in* (3.1) *is an isomorphism from* $\mathbb{R}^{[1]}$ *to* $V_k$, *its range* $V_k$ *is* $T^{[k]}$*-invariant, and* $\psi T = T^{[k]}\psi$. *Furthermore,* $T^{[k]}$ *is nilpotent on* $\mathbb{R}^{[k]}/V_k$, *so* $V_k$ *contains the eventual range of* $T^{[k]}$.

*Proof.* Since the sets $\beta^{-(k-1)}(i) \subset \mathscr{S}^{[k]}$ are disjoint for $i \in \mathscr{S}$, it follows that $\psi$ is injective. Recalling our convention that matrices act on the right, we compute the action of $T^{[k]}\psi$ on a basis vector $e_i^{[1]}$ for $i \in \mathscr{S}$ by

$$\begin{aligned} T^{[k]}\psi(e_i^{[1]}) &= T^{[k]}\left(\sum_{\{A: \beta^{k-1}A = i\}} e_A^{[k]}\right) = \sum_{\{A: \beta^{k-1}A = i\}} e_A^{[k]} T^{[k]} \\ &= \sum_{\{A: \beta^{k-1}A = i\}} \sum_{\{B: \beta B = \eta A\}} e_B^{[k]} = \sum_{\{C \in \mathscr{E}^{[k]} : \beta^k C = i\}} e_{\eta C}^{[k]}. \end{aligned}$$

Similarly,

$$\begin{aligned} \psi T(e_i^{[1]}) &= \psi(e_i^{[1]} T) = \psi\left(\sum_{j \in \mathscr{S}} T_{ij} e_j^{[1]}\right) \\ &= \sum_{j \in \mathscr{S}} \sum_{\{A: \beta^{k-1}A = j\}} T_{ij} e_A^{[k]} = \sum_{\{C \in \mathscr{E}^{[k]} : \beta^k C = i\}} e_{\eta C}^{[k]}. \end{aligned}$$

Thus $\psi T = T^{[k]}\psi$, which also shows $V_k$ is $T^{[k]}$-invariant. To show that $T^{[k]}$ is nilpotent on $\mathbb{R}^{[k]}/V_k$, let $A \in \mathscr{S}^{[k]}$ have terminal state $j = \eta^{k-1}A$. Then

$$e_A^{[k]}(T^{[k]})^{k-1} = \sum_{\{B \in \mathscr{S}^{[k]} : \beta^{k-1}B = \eta^{k-1}A\}} e_B^{[k]} = \psi(e_j^{[1]}) \in V_k. \qquad (3.2) \quad \square$$

Suppose $B = \xi_1 \cdots \xi_k$, and that $j = \eta(\xi_{k-1})$ has the property that $\sum_{l=1}^{r} T_{jl} = 1$. Then the subblock $B' = \xi_1 \cdots \xi_{k-1}$ determines $B$, and forbidding $B'$ is the same as forbidding $B$. Call a block *reduced* if no proper subblock determines it. We shall prove our results for reduced blocks. By irreducibility of $T$, every block of length $k > 2r$ contains a reduced subblock of length greater than $k - 2r$ that determines it. Hence most of our results hold for all blocks of length $k$, perhaps with different constants, by applying them to this reduced subblock.

Let $B \in \mathscr{B}_k(X_T)$ be reduced. We will enlarge $V_k$ by $k - 1$ dimensions to obtain a $T\langle B\rangle$-invariant subspace $W_k = W_{k,B}$ containing the nonnilpotent part of $T\langle B\rangle$ and calculate the matrix of $T\langle B\rangle$ with respect to a natural basis of $W_k$.

For notational simplicity, let $U$ denote $T^{[k]}$, and $E = E_{\beta B, \eta B}$, so $T\langle B\rangle = U - E$. Also, let $e = e_{\eta B}^{[k]}$, so the range of $E$ is $\mathbb{R}e$. Since $B$ is reduced, it follows that $e \notin V_k$, since otherwise the proper subblock $\beta^{k-1}B$ would determine $B$.

The next result shows that the absorption time of $e$ into $V_k$ under $U$ is $k - 1$.

LEMMA 2. *With the above notations*, $\min\{m > 0 : eU^m \in V_k\} = k - 1$.

*Proof.* By (3.2) the minimum is at most $k - 1$.

Equality is proven by observing that the next to last state $i$ in $B$ must be followed by at least two symbols since $B$ is reduced. Hence, the set of blocks in $\mathscr{S}^{[k]}$ whose first symbol is the last symbol of $B$ is a proper subset of the set of those whose initial state is $i$.

Let $\xi = \eta^{k-1}B$ be the terminal symbol of $B$, and $i = \beta\eta^{k-1}B$ its initial state. Since $B$ is reduced, $\sum_{j \in \mathscr{S}} T_{ij} \geqq 2$. Hence

$$eU^{k-2} = e_{\eta B}^{[k]}(T^{[k]})^{k-2} = \sum_{\{C : \beta^{k-2}C = \xi\}} e_C^{[k]}$$

is summed over a proper subset of $\beta^{-k+1}(i)$. For fixed $i$, every vector in $V_k$ must have the same coordinate on each $e_C^{[k]}$ for $C \in \beta^{-k+1}(i)$. Hence $eU^{k-2} \notin V_k$. $\square$

LEMMA 3. *With the above notations, the vectors* $e, eU, \cdots, eU^{k-2}$ *are linearly independent of each other and of* $V_k$.

*Proof.* Suppose $a_1 e + a_2 eU + \cdots + a_{k-1}eU^{k-2} + v = 0$ for some $v \in V_k$. If all $a_j = 0$, we are done. If not, choose $j$ minimal so $a_j \neq 0$. Applying $U^{k-j-1}$ on the right gives

$$a_j eU^{k-2} = v - (a_{j+1}eU^{k-1} + \cdots + a_{k-1}eU^{2k-3-j}) \in V_k,$$

contradicting Lemma 2. $\square$

In view of Lemma 3, we can introduce the subspace

$$W_k = W_{k,B} = V_k \oplus \mathbb{R}e \oplus \mathbb{R}eU \oplus \cdots \oplus \mathbb{R}eU^{k-2} \subset \mathbb{R}^{[k]},$$

having dimension $r + k - 1$, where $r = \dim V_k = \dim \mathbb{R}^{[1]}$ is the size of $T$. Although $V_k$ depends only on $k$, note that $W_k$ also depends on $B$.

LEMMA 4. *The subspace $W_k$ is invariant under both $U$ and $U - E$, and it contains the eventual range of both transformations.*

*Proof.* The invariance of $W_k$ under $U$ is clear. Since the range of $E$ is $\mathbb{R}e$, it follows that $W_k$ is also invariant under $U - E$. Since by Lemma 1 we have $U$ nilpotent on

$\mathbb{R}^{[k]}/W_k$, and $E$ vanishes there, $U-E$ is nilpotent on $\mathbb{R}^{[k]}/W_k$, so the eventual range of $U-E$ is contained in $W_k$. □

If $\psi$ is the map defined by (3.1), by Lemma 1 the set $\{\psi(e_i^{[1]}) : i \in \mathcal{S}\}$ forms a basis for $V_k$. Hence

$$\mathbb{B} = \{\psi(e_1^{[1]}), \cdots, \psi(e_r^{[1]}), e, eU, \cdots, eU^{k-2}\}$$

is a basis for $W_k$.

To describe the matrix of $U-E$ with respect to $\mathbb{B}$, let $\phi_B(t) = \sum_{j=1}^{k} c_j t^{j-1}$ be the correlation polynomial of $B = \xi_1 \cdots \xi_k$, where $c_j = 1$ if $\xi_1 \cdots \xi_j = \xi_{k-j+1} \cdots \xi_k$ and $c_j = 0$ otherwise. Note that $c_k = 1$, so deg $\phi_B(t) = k-1$. Define

$$C(\phi_B) = \begin{bmatrix} -c_{k-1} & 1 & 0 & \cdots & 0 \\ -c_{k-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -c_2 & 0 & 0 & \cdots & 1 \\ -c_1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

to be its companion matrix. Our definition differs from the usual one by permuting the rows and columns with the permutation $i \to k-i$, but the characteristic polynomial is still $\phi_B(t)$. Let $e_j$ be the $1 \times r$ row vector $[0 \cdots 0\ 1\ 0 \cdots 0]$, and $e_i^T$ be the transpose of $e_i$. Denote the $m \times n$ zero matrix by $0_{m,n}$.

LEMMA 5. *Let $B \in \mathcal{B}_k(X_T)$ be a reduced block with initial state $i = \beta^k B$ and terminal state $j = \eta^k B$. Then the matrix of $T\langle B\rangle = U - E$ with respect to $\mathbb{B}$ is*

$$M_{\mathbb{B}}(U-E) = \left[\begin{array}{c|c} T & -e_i^T \quad 0_{r,k-2} \\ \hline \begin{matrix} 0_{k-2,r} \\ e_j \end{matrix} & C(\phi_B) \end{array}\right].$$

*Proof.* We first compute $M_{\mathbb{B}}(U)$. By Lemma 1, on $V_k$ the matrix of $U$ with respect to the $\psi(e_j^{[1]})$ is just $T$. Also, by (3.1) and (3.2), $(eU^{k-2})U = \psi(e_j^{[1]})$. Hence if

$$(3.3) \qquad J_0(k-1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

denotes the Jordan nilpotent block of size $k-1$, we have

$$M_{\mathbb{B}}(U) = \left[\begin{array}{c|c} T & 0_{r,k-1} \\ \hline \begin{matrix} 0_{k-2,r} \\ e_j \end{matrix} & J_0(k-1) \end{array}\right].$$

Next we compute $M_{\mathbb{B}}(E)$. Using (3.1), we see for $m \in \mathcal{S}$ that

$$\psi(e_m^{[1]})E_{\beta B,\eta B} = \begin{cases} e_{\eta B} & \text{if } m = i = \beta^k B, \\ 0 & \text{otherwise,} \end{cases}$$

and for $0 \leqq m \leqq k-2$ that

$$(eU^m)E_{\beta B,\eta B}=\begin{cases} e_{\eta B}^{[k]} & \text{if } \eta B \text{ and } \beta B \text{ overlap in } k-m-1 \text{ symbols,} \\ 0 & \text{otherwise.}\end{cases}$$

Thus

$$M_{\mathbb{B}}(E)=\left[\begin{array}{c|cc} 0_{r,r} & e_i^T & 0_{r,k-2} \\ \hline & c_{k-1} & \\ & c_{k-2} & \\ 0_{k-1,r} & \vdots & 0_{k-1,k-2} \\ & c_1 & \end{array}\right].$$

Subtraction gives the result. □

It is perhaps interesting to note that although $U-E$ is nonnegative integral, we have represented its "interesting part" by a far smaller matrix $M_{\mathbb{B}}(U-E)$ containing negative entries.

**4. Zeta functions.** We will use the results of the previous section to compute the zeta function of $T\langle B\rangle$ in terms of $\chi_T(t)$, $\phi_B(t)$, and $\operatorname{cof}_{ij}(tI-T)$ described in § 1. Recall our notation from § 1 that $f(t)^\times$ denotes $f(t)$ multiplied by the unique power of $t$ making the product a polynomial with nonzero constant term.

THEOREM 1. *If $B \in \mathscr{B}_k(X_T)$ is reduced with initial state $i=\beta^k B$ and terminal state $j=\eta^k B$, then*

$$\zeta_{T\langle B\rangle}(t)=\frac{1}{[\chi_T(t^{-1})\phi_B(t^{-1})+\operatorname{cof}_{ij}(t^{-1}I-T)]^\times}.$$

COROLLARY 1. *If $T=[n]$, then for every $B \in \mathscr{B}_k(X_T)$ we have*

$$\zeta_{T\langle B\rangle}(t)=\frac{1}{(1-nt)\phi_B(t^{-1})t^{k-1}+t^k}.$$

*In particular, if $B$ has only the trivial overlap with itself, then*

$$\zeta_{T\langle B\rangle}(t)=\frac{1}{1-nt+t^k}.$$

*Proof of Corollary* 1. Since $r=1$, here $\operatorname{cof}_{11}(tI-T)=1$, and the constant term of

$$(t-n)\phi_B(t)+\operatorname{cof}_{11}(tI-T)$$

is $n\phi_B(0)\pm 1 \neq 0$ since $n \geqq 2$ and $\phi_B(0)=0$ or $1$. Hence

$$\begin{aligned}[(t^{-1}-n)\phi_B(t^{-1})+1]^\times &= t^k[(t^{-1}-n)\phi_B(t^{-1})+1] \\ &= (1-nt)\phi_B(t^{-1})t^{k-1}+t^k.\end{aligned}$$

If $B$ overlaps itself only in the entire block, then $\phi_B(t)=t^{k-1}$, completing the proof. □

*Proof of Theorem* 1. Recall from § 3 our notations that $U=T^{[k]}$, $E=E_{ij}$, $T\langle B\rangle=U-E$, $e=e_{\eta B}$, and that $T\langle B\rangle$ is nilpotent on $\mathbb{R}^{[k]}/W_{k,B}$. The matrix of $T\langle B\rangle$ on $W_{k,B}$ is given in Lemma 5, and expansion by the last row shows

$$\chi_{T\langle B\rangle|W_{k,B}}(t)=\chi_T(t)\phi_B(t)+\operatorname{cof}_{ij}(tI-T). \tag{4.1}$$

Applying [4, Thm. 1] finishes the proof. □

Bowen and Lanford [4, p. 45] have pointed out that although deg $\zeta^{-1}_{T\langle B\rangle}(t)$ for full shifts is linear in $|B|$, the straightforward calculation of $\chi_{T\langle B\rangle}(t)$ is exponential in $|B|$, and they asked whether there is a more efficient algorithm. Our reduction of $\mathbb{R}^{[k]}$ to $W_k$ provides such an algorithm. The obvious method of computing $\phi_B(t)$ is quadratic in $|B|$, so Theorem 1 provides a computation of $\zeta_{T\langle B\rangle}(t)$ that is also quadratic in $|B|$. However, a slight modification of one part of the Knuth–Morris–Pratt pattern matching algorithm [13] actually gives a linear algorithm. (We are indebted to Andrew Odlyzko for suggesting this possibility.)

COROLLARY 2. *For fixed $T$ there is an algorithm to compute $\zeta_{T\langle B\rangle}(t)$ in time that is linear in $|B|$.*

*Proof.* By Theorem 1, we need only find a linear algorithm to compute $\phi_B(t)$. Let $B = \xi_1 \cdots \xi_k$. Define $f\colon \{1, \cdots, k\} \to \{0, \cdots, k-1\}$ by setting $f(j)$ to be the largest nonnegative $i < j$ such that $\xi_1 \cdots \xi_i = \xi_{j-i+1} \cdots \xi_j$. As in [13, § 2], the table of values of $f$ can be computed in $O(k)$ steps. Our definition of $f$ differs slightly from that in [13], but this has no consequence for the algorithm.

Since $f$ is strictly decreasing, there is a first iterate $f^s(k)$ that equals zero. Let $k_0 = k$, $k_1 = f(k)$, $k_2 = f(f(k))$, $\cdots$, $k_{s-1} = f^{s-1}(k)$. We claim

$$\phi_B(t) = \sum_{j=0}^{s-1} t^{k_j - 1}.$$

For by definition, $k_1 = f(k)$ records the largest nontrivial overlap of $B$ with itself. It follows that the next largest overlap of $B$ with itself coincides with the largest nontrivial overlap of $\xi_1 \cdots \xi_{k_1}$ with itself, which by definition is $f(k_1) = k_2$. This continues until $f^s(k)$ becomes zero. □

Since the number of $k$-blocks is roughly $\lambda_T^k$, one might expect the number of distinct $\zeta_{T\langle B\rangle}(t)$ to also have exponential growth. Surprisingly, this number turns out to be quite small. For $k = 20$ there are only 116 distinct correlations, and even for $k = 50$ there are only 2,240 [9, p. 29].

COROLLARY 3. *If $\gamma > \frac{1}{2}\log(\frac{3}{2})$, then*

$$|\{\zeta_{T\langle B\rangle}(t) : B \in \mathscr{B}_k(X_T)\}| = O(k^{\gamma \log k}).$$

*Proof.* By [9, Thm. 6.1],

$$|\{\phi_B(t) : B \in \mathscr{B}_k(X_T)\}| = O(k^{\gamma \log k}).$$

Since every block in $\mathscr{B}_k(X_T)$ contains a reduced subblock of length greater than $k - 2r$ determining it, and the number of the polynomials $\operatorname{cof}_{ij}(tI - T)$ is uniformly bounded, the result follows. □

Another curious fact also follows.

COROLLARY 4. *Suppose $m, n \geqq 2$. The number of distinct zeta functions produced by omitting the blocks of length $k$ from the full $m$-shift coincides with that from the full $n$-shift.*

*Proof.* This follows from Corollary 1 together with the result [9, Cor. 5.1] that the set of correlations is independent of the size of the alphabet. □

Because of the possible cancellation of lower powers of $t$ in (4.1), Theorem 1 does not immediately yield deg $\zeta^{-1}_{T\langle B\rangle}(t)$. In particular, a good lower bound is not obvious.

THEOREM 2. *If $B \in \mathscr{B}_k(X_T)$ is reduced with initial state $i$ and terminal state $j$, then*

$$k - 4r \leqq \deg \zeta^{-1}_{T\langle B\rangle}(t) \leqq k + r - 1.$$

*The upper bound is attained if and only if*

$$(\det T)\phi_B(0)+\operatorname{cof}_{ij}(-T)\neq 0, \tag{4.2}$$

*and this occurs for all blocks in a full shift.*

*Proof.* Using the notations from the proof of Theorem 1, we see that

$$\deg \chi_{T\langle B\rangle | W_k}(t)=\dim W_k=k+r-1,$$

and that $T\langle B\rangle$ is invertible on $W_k$ if and only if the constant term in (4.1)

$$\chi_T(0)\phi_B(0)+\operatorname{cof}_{ij}(-T)=(\det T)\phi_B(0)+\operatorname{cof}_{ij}(-T)\neq 0.$$

For full shifts, the proof of Corollary 1 shows the constant term is nonzero for all blocks, completing the proof of the upper bound statements.

For the lower bound, we will prove by induction on $k$ that the highest power $P$ of $t$ dividing (4.1) is less than $5r$. Since the degree is $k+r-1$, the lower bound will follow. For notational simplicity, put $c_{ij}(t)=\operatorname{cof}_{ij}(tI-T)$. A crucial fact that we use repeatedly is that $c_{ij}(\lambda_T)\neq 0$ for all $i,j$ [16, p. 7].

If $k\leqq 4r$, then for every $B\in\mathscr{B}_k(X_T)$ the degree of (4.1) is $<5r$. Also, at $t=\lambda_T$ this polynomial has value $\operatorname{cof}_{ij}(\lambda_T I-T)\neq 0$ by [16, p. 7]. Hence if $k\leqq 4r$, the highest power $P$ of $t$ dividing (4.1) has $P<5r$.

Now fix $k>4r$, and assume our inductive hypothesis for all blocks of length $<k$. Let $B\in\mathscr{B}_k(X_T)$. Choose $p\geqq 1$ minimal so $B$ overlaps itself in $k-p$ symbols. If $B$ has only trivial overlaps, define $p$ to be $k$. This $p$ is the *fundamental* period of $B$ as defined in [9]. We distinguish two cases, depending on the relationship of $p$ to $r$.

First suppose $p>r$. If $p=k$ the result is trivial since then $\phi_B(t)=t^{k-1}$, $\deg c_{ij}(t)\leqq r-1$, and $c_{ij}(\lambda_T)\neq 0$, so $P\leqq r-1$. If $p<k$, let $C=\eta^pB$. Then $C$ has initial state $i$, terminal state $j$, and $\phi_B(t)=t^{k-1}+\phi_C(t)$ with $\deg\phi_C(t)=k-p-1$. Since $\deg\chi_T(t)=r$, and $p>r$, the highest power of $t$ dividing (4.1) coincides with the highest power dividing $\chi_T(t)\phi_C(t)+c_{ij}(t)$, which is less than $5r$ by induction.

Next suppose $p\leqq r$. We first treat the case $t^p-1\nmid\chi_T(t)$. By the recursive description of $\phi_B(t)$ in [9, Thm. 5.1], it follows that

$$\phi_B(t)=\sum_{m=0}^{\lfloor (k-p-1)/p\rfloor} t^{k-1-mp}+\psi(t), \tag{4.3}$$

where $\deg\psi(t)<2p$. We claim that for every $K\geqq 1$, in the product

$$(t^{pK}+t^{p(K-1)}+\cdots+t^p+1)\chi_T(t)$$

every set of $p$ consecutive powers of $t$ in $[0,Kp+r]$ has at least one nonzero coefficient. For if not, then letting $|t|<1$ and $K\to\infty$ shows that $(t^p-1)^{-1}\chi_T(t)\in\mathbb{Z}[t]$, contradicting $t^p-1\nmid\chi_T(t)$. Substituting (4.3) into (4.1) and applying our claim, it follows that (4.1) has at least one nonzero coefficient of a power of $t$ in the range $[\max\{r,2p\}+1,\max\{r,2p\}+p]$, showing that $P\leqq 3r$ in this case.

Finally, suppose $p\leqq r$ and $t^p-1\,|\,\chi_T(t)$. By (4.3),

$$\phi_B(t)=t^{p+l}\left(\frac{t^{pK}-1}{t^p-1}\right)+\psi(t),$$

where $K=\lfloor(k-p-1)/p\rfloor$, $l<p$, and $\deg\psi(t)<2p$. Let $\chi_T(t)=(t^p-1)\tilde{\chi}(t)$, so $\tilde{\chi}(\lambda_T)=0$. Then

$$\chi_T(t)\phi_B(t)+c_{ij}(t)=\tilde{\chi}(t)t^{p(K+1)+l}+\tilde{\chi}(t)[(t^p-1)\psi(t)-t^{p+l}]+c_{ij}(t).$$

If $\rho(t) = \tilde{\chi}(t)[(t^p - 1)\psi(t) - t^{p+l}] + c_{ij}(t)$, then $\rho(\lambda_T) = c_{ij}(\lambda_T) \neq 0$, so $\rho(t)$ is not identically zero. Also, $\deg \rho(t) < r + 2p \leqq 3r$. Since

$$p(K+1)+l \geqq p\left(\frac{k-1}{p}-1\right)=k-p-1 \geqq 3r,$$

the highest power $P$ of $t$ dividing (4.1) coincides with that dividing $\rho(t)$, so $P \leqq 3r$ in this case. □

**5. Entropy.** How does entropy change when one long block is removed? Since $h(\sigma_T) = \log \lambda_T$, this amounts to determining the change in spectral radius. Standard Perron–Frobenius theory [16, Thm. 1.1(*e*)] shows $\lambda_T > \lambda_{T\langle B\rangle}$. The following considerations allow us to guess that the difference is exponentially small in $|B|$. We can consider $T\langle B\rangle$ to be the end result of $U - tE$ as $t$ varies from 0 to 1. To find the change in eigenvalue, it becomes important to know its rate of change, say at $t = 0$. Surprisingly, this rate is a simple function of the left and right eigenvectors.

LEMMA 6. *Let $A = [a_{ij}]$ be a real square matrix with simple eigenvalue $\lambda$ and corresponding left eigenvector $v$ and right eigenvector $w$. Then*

$$\left[\frac{\partial \lambda}{\partial a_{ij}}\right]=\frac{wv}{vw}.$$

*Proof.* Since $\lambda$ is simple, it is known [12, Thm. II.1.8] that $\lambda$, $v$, and $w$ are analytic functions of $a_{ij}$. Take $\partial/\partial_{ij}$ of $Aw = \lambda w$, use $\partial A/\partial a_{ij} = E_{ij}$, and multiply by $v$ on the left to obtain

$$vA\frac{\partial w}{\partial a_{ij}}+vE_{ij}w=v\frac{\partial \lambda}{\partial a_{ij}}w+\lambda v\frac{\partial w}{\partial a_{ij}}.$$

Cancelling the terms involving $vA = \lambda v$ shows that $v_i w_j = (vw)(\partial\lambda/\partial a_{ij})$. Since $\lambda$ is simple, $vw \neq 0$, concluding the proof. □

Let the notation $f(k) \asymp g(k)$ as $k \to \infty$ mean that there are $a$, $b > 0$ so that for sufficiently large $k$ we have $ag(k) < f(k) < bg(k)$.

Now fix $T$, and let $v > 0$ and $w > 0$ be the left and right eigenvectors for $\lambda_T$. Put $c = \min\{v_i, w_i\}$ and $d = \max\{v_i, w_i\}$. Recalling the map $\psi$ defined in (3.1), we have that $\psi(v)$, $\psi(w)$ are left and right eigenvectors for $U = T^{[k]}$, and have entries between $c$ and $d$. Since these vectors have length about $\lambda_T^k$, it follows $\psi(v)\psi(w) \asymp \lambda_T^k$. Hence by Lemma 6, the derivative of $\lambda_{U-tE}$ at $t = 0$ is within two positive constants of $\lambda_T^{-k}$. If we were able to extend this estimate of the derivative to all $0 \leqq t \leqq 1$, we would be done. However, the estimate runs into trouble as $t$ becomes positive since the eigenvectors also vary with $t$.

Nevertheless, the linear algebra of § 3 enables us to prove a sharp exponential estimate.

THEOREM 3. *There are constants $c_T$, $d_T > 0$ so that if $k$ is sufficiently large, for every $B \in \mathscr{B}_k(X_T)$ we have*

$$c_T\lambda_T^{-k} < h(\sigma_T) - h(\sigma_{T\langle B\rangle}) < d_T\lambda_T^{-k}.$$

*Proof.* In the following, the $a_j$ are suitable positive constants. Let us first consider the case $T = [n]$ of the full $n$-shift. Then $\lambda_T = n$ and $\lambda_{T\langle B\rangle}$ satisfies $(t - n)\phi_B(t) + 1 = 0$. By [8, Lemma 3], $|\phi_B(z)| > a_1(1.7)^k$ for $|z| \geqq 1.7$ and suitable $a_1 > 0$. Thus if $|z| = 1.7$ we have

$$|(z-n)\phi_B(z)| \geqq (0.3)a_1(1.7)^k > 1 \tag{5.1}$$

for $k$ large enough. We use this to count roots by invoking the following result from complex analysis.

ROUCHÉ'S THEOREM. *Let $f(z)$ and $g(z)$ be analytic on $\{|z| \leqq R\}$ and satisfy $|f(z) - g(z)| < |f(z)|$ for $|z| = R$. Then $f(z)$ and $g(z)$ have the same number of zeros in $\{|z| < R\}$.*

Using (5.1) and Rouché's Theorem, it follows that $(z-n)\phi_B(z)$ and $(z-n)\phi_B(z) + 1$ have the same number of roots in $|z| \geqq 1.7$, namely 1. This shows that $\lambda = \lambda_{T\langle B\rangle} \geqq 1.7$. Hence

$$\frac{1}{\lambda^{k-1}+\lambda^{k-2}+\cdots+1} \leqq |n-\lambda| = \frac{1}{\phi_B(\lambda)} = \frac{1}{\lambda^{k-1}+c_{k-2}\lambda^{k-2}+\cdots+c_0} \leqq \frac{1}{\lambda^k}.$$

Thus

$$|n-\lambda_{T\langle B\rangle}| \asymp \lambda_{T\langle B\rangle}^{-k} \quad \text{for } B \in \mathscr{B}_k(X_T) \text{ as } k \to \infty.$$

We claim that actually $|n - \lambda_{T\langle B\rangle}| \asymp n^{-k}$, for which it suffices to show that $|n - \lambda_{T\langle B\rangle}| \leqq a_2 n^{-k}$ for suitable $a_2 > 0$ since $\lambda = \lambda_{T\langle B\rangle} < n$. Now

$$|n-\lambda| \leqq a_3\lambda^{-k} = a_3 n^{-k\log\lambda/\log n},$$

and since $|n-\lambda| < a_4\lambda^{-k}$, it follows, using differentiability of $\log x$ at $x = n$, that $-\log\lambda/\log n < -1 + a_5\lambda^{-k} < -1 + a_5(1.7)^{-k}$. Hence

$$|n-\lambda| \leqq a_3 n^{-k} n^{a_5 k(1.7)^{-k}}.$$

Putting

$$a_2 = a_3 \min_{1\leqq k<\infty} \{n^{a_5 k(1.7)^{-k}}\} < \infty$$

completes the proof for the full shift.

To extend these ideas to general $T$, we need a version of (5.1) that works for smaller $|z|$.

LEMMA 7. *Fix $\rho > 1$. Then*

$$\inf_{B\in\mathscr{B}_k(X_T)} \inf_{|z|\geqq\rho} |\phi_B(z)| \to \infty \quad \textit{as } k \to \infty.$$

*Proof.* Recall from the proof of Theorem 2 that if $p$ is the fundamental period for $B$, then

$$\phi_B(t) = \sum_{m=0}^{\lfloor (k-p-1)/p\rfloor} t^{k-1-mp} + \psi(t),$$

where $\deg\psi(t) < 2p$, and $\psi(t)$ has coefficients that are 0 or 1. Fix $M > 0$. If $p > k/10$, then for all $|z| \geqq \rho$ we have

$$|\phi_B(z)| \geqq |z|^{k-1} - 2\sum_{m=0}^{\lfloor 9k/10\rfloor} |z|^m \geqq M$$

provided $k$ is large enough. If $p \leqq k/10$, then for $K = \lfloor (k-1)/p \rfloor$ we have

$$\phi_B(t) = t^l\left(\frac{t^{Kp}-1}{t^p-1}\right) + \psi(t),$$

where $l < p$ and $\deg \psi(t) < 2p$. Hence for all $|z| > \rho$ we have that

$$|\phi_B(z)| \geqq \left|z^l\left(\frac{z^{Kp}-1}{z^p-1}\right)\right| - \sum_{m=0}^{2p} |z|^m$$

$$\geqq \frac{|z|^{9k/10} - |z|^{k/10}}{|z|^{k/10}+1} - \frac{k}{5}|z|^{k/5} \geqq M$$

if $k$ is sufficiently large. □

We now prove Theorem 3 for general $T$. We know that $\lambda = \lambda_{T\langle B\rangle}$ satisfies

$$\chi_T(t)\phi_B(t) + c_{ij}(t) = 0, \tag{5.2}$$

where $c_{ij}(t) = \operatorname{cof}_{ij}(tI - T)$ is one of a finite set of polynomials. Since $\chi_T(t) = (t - \lambda_T)q(t)$, where $q(\lambda_T) \neq 0$, we have

$$|\lambda_T - t| = \frac{|c_{ij}(t)|}{|\phi_B(t)q(t)|}.$$

Let the roots of $\chi_T(t)$ be $\lambda_1 = \lambda_T, \lambda_2, \cdots, \lambda_r$. Fix $\rho$ with $\lambda_T > \rho > \max_{2 \leqq j \leqq r}|\lambda_j|$. Apply Lemma 7 to conclude by Rouché's Theorem that for sufficiently large $k$ (5.2) has exactly one solution for $|t| > \rho$, namely $t = \lambda_{T\langle B\rangle}$. Hence

$$|\lambda_T - \lambda_{T\langle B\rangle}| = \frac{|c_{ij}(\lambda_{T\langle B\rangle})|}{|\phi_B(\lambda_{T\langle B\rangle})q(\lambda_{T\langle B\rangle})|} \to 0 \quad \text{as } k \to \infty.$$

Since $q(z) \neq 0$ if $|z| > \rho$, we have $|q(\lambda_{T\langle B\rangle})| \asymp 1$ as $k \to \infty$. Furthermore, as remarked above, the Perron–Frobenius theory shows that $c_{ij}(\lambda_T) \neq 0$ for all $i, j$, so $|c_{ij}(\lambda_{T\langle B\rangle})| \asymp 1$ as $k \to \infty$. Since $|\phi_B(\lambda_{T\langle B\rangle})| \asymp \lambda^k_{T\langle B\rangle}$, we obtain

$$|\lambda_T - \lambda_{T\langle B\rangle}| \asymp \lambda^{-k}_{T\langle B\rangle}.$$

The conclusion that this forces

$$|\lambda_T - \lambda_{T\langle B\rangle}| \asymp \lambda_T^{-k} \quad \text{as } k \to \infty$$

follows exactly as in the full shift case. □

We remark that these estimates may also be proved using generating functions, in the spirit of the calculations in [8, § 2] for the full shift. Let $f_n^{(ij)}$ be the number of blocks in $\mathscr{B}_n(X_T)$ from $i$ to $j$ not containing $B$, and put $F_{ij}(z) = \sum_{n=0}^{\infty} f_n^{(ij)} z^{-n}$, and set $F(z) = [F_{ij}(z)]$. Then developing matrix analogues of the polynomial relations in [8, § 1], we can show that

$$F(z) = (I - z^{-(k-1)}E)[zI - T + \phi_B(z)^{-1}TE]^{-1},$$

where $E = E_{ij}$. The largest root of the denominator is $\lambda_{T\langle B\rangle}$, and we can derive the exponential estimate of Theorem 3 from this. The details are more involved than for the proof given here. This generating function technique could also be used when omitting a finite number of blocks, as in [10, § 2]. This yields a nonsingular system of matrix equations whose solution would allow an estimate of the resulting spectral radius. However, the results are not nearly as explicit as for one block.

We also remark that much finer information about $|\lambda_T - \lambda_{T\langle B\rangle}|$ can be obtained from the proof of Theorem 3. For example, if $T = [n]$ and $B \in \mathscr{B}_k(X_T)$ has only the trivial overlap with itself, then

$$\lambda_{T\langle B\rangle} = n - \frac{1}{n^{k-1}} - \frac{k-1}{n^{2k-1}} + O\left(\frac{k^2}{n^{3k}}\right).$$

Such estimates can be proved directly [8, Lemma 4], or with the Lagrange inversion formula [17, Thm. 2].

**6. Jordan forms.** Let $J^\times(T)$ be the invertible part of the Jordan form $J(T)$ for $T$, and $J^0(T)$ be its nilpotent part. Williams [18] proved that $J^\times(T)$ is a topological invariant of $\sigma_T$. Although $J^0(T)$ is not an invariant, our analysis allows us to compute $J^0(T^{[k]})$ and, in many cases, $J^0(T\langle B\rangle)$.

Denote by $J_0(m)$ the elementary $m \times m$ Jordan block for eigenvalue 0 as in (3.3), and by $J_0(m)^{\oplus n}$ the direct sum of $n$ copies of $J_0(m)$. We first show that in passing from $T$ to $T^{[2]}$, each Jordan nilpotent block of $J^0(T)$ increases by one dimension, and enough one-dimensional nilpotent blocks are added to account for the rest of dim $\mathbb{R}^{[2]}$.

LEMMA 8. *Suppose $T$ has Jordan form $J^\times(T) \oplus J^0(T)$, where*

$$J^0(T) = J_0(1)^{\oplus n_0} \oplus J_0(2)^{\oplus n_1} \oplus \cdots \oplus J_0(p)^{\oplus n_{p-1}}.$$

*Then $T^{[2]}$ has Jordan form $J^\times(T) \oplus J^0(T^{[2]})$, where*

$$J^0(T^{[2]}) = J_0(1)^{\oplus n_{-1}} \oplus J_0(2)^{\oplus n_0} \oplus \cdots \oplus J_0(p+1)^{\oplus n_{p-1}},$$

*and*

$$n_{-1} = \sum_{i,j=1}^{r} T_{ij} - \dim J^\times(T) - \sum_{l=1}^{p} (l+1) n_{l-1}. \tag{6.1}$$

*Proof.* Since $T$ is assumed irreducible, for each $i \in \mathcal{S}$ we can choose $\xi(i) \in \mathcal{E}$ with $\eta(\xi(i)) = i$. Define $g\colon \mathbb{R}^{[1]} \to \mathbb{R}^{[2]}$ by $g(\sum_{i=1}^r a_i e_i^{[1]}) = \sum_{i=1}^r a_i e_{\xi(i)}^{[2]}$. Recalling the map $\psi$ defined in (3.1), note that $T^{[2]} \circ g = \psi$. If $\{v_1, \cdots, v_p\}$ is a Jordan basis for $J^\times(T)$, then $\{\psi(v_1), \cdots, \psi(v_p)\}$ is one for $J^\times(T^{[2]})$ by Lemma 1. Let $V$ denote the span of the $\psi(v_m)$. Suppose $w \in \mathbb{R}^{[1]}$ generates one of the Jordan nilpotent blocks of size $s$ in $J^0(T)$. Then $g(w)$ generates a Jordan nilpotent block of size $s+1$ in $J^0(T^{[2]})$ since $g(w)T^{[2]} = \psi(w)$. Thus by mapping each Jordan nilpotent generator in $J^0(T)$ to its image under $g$, we obtain vectors in $\mathbb{R}^{[2]}$ with nilpotency $\geqq 2$ under $T^{[2]}$. On the subspace $W$ generated by powers of $T^{[2]}$ on these vectors the matrix of $T^{[2]}$ is

$$J_0(2)^{\oplus n_0} \oplus J_0(3)^{\oplus n_1} \oplus \cdots \oplus J_0(p+1)^{\oplus n_{p-1}}.$$

For each $i \in \mathcal{S}$ let

$$\mathcal{S}(i) = \{\xi \in \mathcal{E} : \eta(\xi) = i, \xi \neq \xi(i)\}.$$

The vectors $\{e_\xi^{[2]} - e_{\xi(i)}^{[2]} : \xi \in \mathcal{S}(i), i \in \mathcal{S}\}$ are each annihilated by $T^{[2]}$ and span the remaining part of $\mathbb{R}^{[2]}$. It follows as in the proof of the Jordan form [10, § 7.3] that $V \oplus W$ has a $T^{[2]}$-invariant complement on which $T^{[2]}$ is 0. Since $\dim \mathbb{R}^{[2]} = \sum_{ij} T_{ij}$, the dimension $n_{-1}$ of this complement is given in (6.1), concluding the proof. □

Since $T^{[k]} \cong (T^{[k-1]})^{[2]}$, Lemma 8 allows the inductive determination of $J^0(T^{[k]})$. It is convenient to introduce the sequence $\{n_q(T) : q \in \mathbb{Z}\}$ of integers defined as follows. For $q \geqq 0$ the $n_q(T)$ are determined by

$$J^0(T) = \bigoplus_{q=0}^{\infty} J_0(q+1)^{\oplus n_q(T)},$$

so $n_q(T) = 0$ for sufficiently large $q$. For $q < 0$ we use the backward recursion

$$n_q(T) = \sum_{i,j=1}^{r} (T^{|q|})_{ij} - \dim J^\times(T) - \sum_{l=1}^{\infty} (l+1) n_{l+q}, \tag{6.2}$$

where the infinite series is really finite since $n_{l+q} = 0$ for large $l$.

THEOREM 4. *If $n_q(T)$ is defined as above, then*

$$J^0(T^{[k]}) = \bigoplus_{q=0}^{\infty} J_0(q+1)^{\oplus n_{q+1-k}(T)}.$$

*Proof.* If $k = 1$ this follows by definition, while Lemma 8 shows this to be true for $k = 2$. The general result follows by induction on $k$. □

For a concrete example, consider the full 2-shift $T = [2]$. The theorem implies that for all $k \geqq 3$ we have

$$J^0(T^{[k]}) = J_0(1)^{\oplus 2^{k-3}} \oplus J_0(2)^{\oplus 2^{k-4}} \oplus \cdots \oplus J_0(k-2)^{\oplus 2^0} \oplus J_0(k-1).$$

Mike Boyle (private communication) has pointed out to us that

$$\operatorname{rk}(T^{[k]})^p = \operatorname{rk}(T^{[k+l]})^{p+l} \tag{6.3}$$

for $l, p \geqq 0$ and $k \geqq 1$. The proof is analogous to that of Lemma 8. If $S$ is any matrix, then the number of Jordan nilpotent blocks of size $m \geqq 1$ is $\operatorname{rk}(S^{m+1}) - 2\operatorname{rk}(S^m) + \operatorname{rk}(S^{m-1})$. Hence (6.3) yields an alternative proof of Theorem 4.

We now show that under some circumstances we can determine the Jordan form of $T\langle B\rangle$.

THEOREM 5. *Suppose $T$ is invertible, that $B \in \mathscr{B}_k(X_T)$ is reduced with initial state $i$ and terminal state $j$, and that*

$$(\det T)\phi_B(0) + \operatorname{cof}_{ij}(-T) \neq 0. \tag{6.4}$$

*This condition is met by all blocks in a full shift. Then $T\langle B\rangle$ is invertible on $W_k$, $J^\times(T\langle B\rangle)$ is just the Jordan form of the restriction of $T\langle B\rangle$ to $W_k$, and $J^0(T\langle B\rangle)$ is obtained from $J^0(T^{[k]})$ by deleting one copy of $J^0(k-1)$.*

*Proof.* Use the notations of § 3, with $U = T^{[k]}$, $e = e^{[k]}_{\eta B}$, $E = E_{\beta B, \eta B}$, and $V_k$ and $W_k$ the subspaces described there. Since $T$ is invertible, there is a $v \in V_k$ such that $vU^{k-1} = eU^{k-1}$. Since $eU^{k-2} \notin V_k$, it follows that $w = e - v$ generates a nilpotent Jordan block under $U$ of maximal size, and that $W_k$ is the direct sum of the corresponding subspace and $V_k$. The proof of the Jordan Theorem [10, § 7.3] shows that there is a Jordan basis for $U$ with $w$ as one of its generating basis vectors. Since $E$ vanishes on the direct complement of $W_k$, it follows that passing from $U$ to $U - E = T\langle B\rangle$ preserves the direct complement Jordan structure. By (6.4), it follows as in Theorem 2 that $U - E$ is invertible on $W_k$. Thus in passing from $U$ to $U - E$, one Jordan nilpotent block of size $k - 1$ combines with $V_k$ to form the subspace $W_k$ on which the invertible part of $U - E$ acts, while the remaining Jordan nilpotent blocks remain undisturbed. □

## REFERENCES

[1] R. ADLER, D. COPPERSMITH, AND M. HASSNER, *Algorithms for sliding block codes*, IEEE Trans. Inform. Theory, 29 (1983), pp. 5–22.

[2] M. ARTIN AND B. MAZUR, *On periodic points*, Ann. of Math., 81 (1965), pp. 82–89.

[3] R. BOWEN, *Equilibrium States and the Ergodic Theory of Anosov Diffeomorphisms*, Lecture Notes Math. 470, Springer-Verlag, New York, 1975.

[4] R. BOWEN AND O. LANFORD III, *Zeta functions of restrictions of the shift transformation*, AMS Proc. Symp. Pure. Math., 14 (1970), pp. 43–49.

[5] M. BOYLE, D. LIND, AND D. RUDOLPH, *The automorphism group of a shift of finite type*, Trans. Amer. Math. Soc., 306 (1988), pp. 71–114.

[6] J. E. DAYHOFF AND G. L. GERSTEIN, *Favored patterns in spike trains*, J. Neurophysiol., 49 (1983), pp. 1334–1363.

[7] E. N. Gilbert, *Synchronization of binary messages*, IRE Trans. Inform. Theory, IT-6 (1960), pp. 470–477.

[8] L. J. Guibas and A. M. Odlyzko, *Maximal prefix-synchronized codes*, SIAM J. Appl. Math., 35 (1978), pp. 401–418.

[9] L. J. Guibas and A. M. Odlyzko, *Periods in strings*, J. Combin. Theory Ser. A, 30 (1981), pp. 19–42.

[10] L. J. Guibas and A. M. Odlyzko, *String overlaps, pattern matching, and non-transitive games*, J. Combin. Theory Ser. A, 30 (1981), pp. 183–208.

[11] K. Hoffman and R. Kunze, *Linear Algebra*, 2nd ed., Prentice-Hall, Englewood Cliffs, 1971.

[12] T. Kato, *A Short Introduction to Perturbation Theory for Linear Operators*, Springer-Verlag, New York, 1982.

[13] D. E. Knuth, J. H. Morris, and V. R. Pratt, *Fast pattern matching in strings*, SIAM J. Comput., 6 (1977), pp. 323–350.

[14] B. Marcus, *Sofic systems and encoding data*, IEEE Trans. Inform. Theory, 31 (1985), pp. 366–377.

[15] W. Parry and S. Tuncel, *Classification Problems in Ergodic Theory*, LMS Lecture Notes 67, Cambridge University Press, Cambridge, 1982.

[16] E. Seneta, *Non-Negative Matrices and Markov Chains*, Springer-Verlag, New York, 1981.

[17] H. S. Wilf, *Strings, substrings, and the "nearest integer" function*, Amer. Math. Monthly, 94 (1987), pp. 855–860.

[18] R. Williams, *Classification of subshifts of finite type*, Ann. of Math., 98 (1973), pp. 120–153; *errata*, 99 (1974) pp. 380–381.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 366–370, August 1989

# IDENTITIES SATISFIED BY ITERATED POLYNOMIALS AND $(Q, x)$-BINOMIAL COEFFICIENTS*

C. L. MALLOWS†

**Abstract.** Lagarias and Reeds showed that iterates $Q^{(i)}(x) = Q(Q^{(i-1)}(x))$ of a polynomial $Q(x)$ satisfy certain identities. It is shown that their result can be interpreted as a symbolic generalization of the result "the $(p + 1)$st-difference of a polynomial of degree $p$ is zero," in which powers of an independent variable are replaced by iterates of a polynomial transformation. $(Q, x)$-binomial coefficients $[{n \atop i}]_{Q,x}$ are defined where $Q$ is a polynomial and $x$ a scalar, which are defined by a two-term recurrence showing that $[{n \atop i}]_{Q,x}$ are in the polynomial ring $\mathbb{Z}[Q, x]$ generated by $x$ and the coefficients of $Q$. For polynomial $P(x)$ of degree $n$, the iterates $Q^{(k)}(x)$ satisfy the identity $P(Q^{(n+1)}(x)) = \sum_{j=0}^{n} (-1)^j [{n \atop j}]_{Q,x} P(Q^{(j)}(x))$. If $Q(x) = qx + a$ is linear, then $[{n \atop j}]_{Q,x} = q^{\binom{n-j}{2}}[{n \atop j}]_q$ where $[{n \atop j}]_q$ is a $q$-binomial coefficient. It is shown by example that other $q$-formulae have $(Q, x)$-analogues.

**Key words.** recurrences, $q$-binomial coefficients

**AMS(MOS) subject classifications.** 11B37, 05A30, 11T41

**1. Iterated polynomials.** We work throughout in a commutative ring $A$, with unit. We consider iterates of a polynomial recurrence: $Q : A \to A$, where $Q$ is a polynomial (of degree $d$, say), with coefficients in $A$. We write

$$Q^{(k)}(x) = Q(Q^{(k-1)}(x))$$

and

$$Q^{(0)}(x) = x, \qquad Q^{(1)}(x) = Q(x).$$

We also consider a second polynomial transformation $P : A \to A$ of degree $p$ and the corresponding sequence of quantities

$$p_k = P(Q^{(k)}(x)) \qquad k = 0, 1, \cdots .$$

Lagarias and Reeds ([3, Thm. 3.2]) established the following identities for such quantities.

THEOREM 1 (Lagarias and Reeds). *There exist coefficients $b_{pj} (j = 0, 1, \cdots p) \in \mathbb{Z}[Q, x]$ (polynomials in $x$ and the coefficients of $Q$, but independent of the coefficients of $P$) such that*

$$(1) \qquad P(Q^{(p+1)}(x)) = \sum_{i=0}^{p} b_{pi} P(Q^{(i)}(x))$$

*as a formal identity.*

The proof in [3] is given only for the case $p = d$ but extends easily to the general case $p \neq d$. They ask ([3, § 1]) whether these identities have an interesting algebraic interpretation. We shall show that (1) is an analogue of the familiar result, valid for all polynomials of degree $\leqq p$,

$$(2) \qquad P(x+p+1) = \sum_{i=0}^{p} (-1)^{p-j} \binom{p+1}{i} P(x+i)$$

which asserts that the $(p + 1)$st-difference of a polynomial of degree $p$ is identically zero.

* Received by the editors June 27, 1988; accepted for publication (in revised form) November 23, 1988.
† AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

**2. The linear case.** Let us first take $Q(x) = x + a$. Then

$$Q^{(k)}(x) = x + ka \qquad k = 0, 1, \cdots$$

so $P(Q^{(k)}(x))$ is a polynomial in $k$, of degree $p$. By (2) we have that (1) holds, with

$$b_{pi} = (-1)^p \binom{p+1}{i}.$$

Now take $Q(x) = qx + a$. Then

$$Q^{(k)}(x) = q^k x + q^{k-1}a + q^{k-2}a + \cdots + a. \tag{3}$$

Following Lagarias and Reeds, we can determine the coefficients $b_{po}, \cdots, b_{pp}$ by solving the system of equations (which they must satisfy, if they exist in $Z(Q, x)$)

$$V_p b_p = c_{p+1}$$

where

$$b_p = (b_{p0}, \cdots, b_{pp})^t, c_{p+1} = (x, Q^{(p+1)}(x), (Q^{(p+1)}(x))^2, \cdots, (Q^{(p+1)}(x))^p)^t \tag{4}$$

and $V_p$ is a Vandermonde matrix with

$$(V_p)_{ij} = (Q^{(j)}(x))^i \qquad 0 \leqq i, j \leqq p.$$

The solution of (4) is, formally,

$$b_{pi}(Q, x) = \prod_{\substack{j=0 \\ j \neq i}}^{p} \frac{Q^{(p+1)}(x) - Q^{(j)}(x)}{Q^{(i)}(x) - Q^{(j)}(x)} \qquad i = 0, 1, \cdots, p, \tag{5}$$

which clearly depends (in general) on both $Q$ and $x$. However, when $Q^{(k)}$ is given by (3), the dependence on $x$ drops away, and

$$b_{pi} = \prod_{\substack{j=0 \\ j \neq i}}^{p} \frac{q^{p+1} - q^j}{q^i - q^j} = (-1)^{p-i} q^{\binom{p-i+1}{2}} \begin{bmatrix} p+1 \\ i \end{bmatrix}_q$$

where $\begin{bmatrix} p+1 \\ i \end{bmatrix}_q$ is a $q$-binomial coefficient, defined for $0 \leqq i \leqq n$ as

$$\begin{bmatrix} n \\ i \end{bmatrix}_q = \frac{(q^{n-i+1})_i}{(q)_i} \qquad \begin{bmatrix} n \\ 0 \end{bmatrix}_q = \begin{bmatrix} n \\ n \end{bmatrix}_q = 1$$

where

$$(a)_i = (1-a)(1-aq)(1-aq^2)\cdots(1-aq^{i-1})$$

(see, for example, [1, p. 15]).

We shall find it convenient to write

$$\binom{n}{i}_q = q^{\binom{n-i}{2}} \begin{bmatrix} n \\ i \end{bmatrix}_q$$

so that

$$b_{pi} = (-1)^{p-i} \binom{p+1}{i}_q.$$

These modified coefficients satisfy the recurrence

$$(6)\qquad \binom{p+1}{i}_q = \binom{p}{i-1}_q + q^p\binom{p}{i}_q$$

which provides an immediate demonstration (by induction) that they are polynomials in $q$.

**3. The general case.** Now we take $Q$ to be a general polynomial of degree $d \geqq 1$. The formal result (5) still holds, and Lagarias and Reeds proved (in the case $p = d$) that it does in fact determine $b_{pi}$ as a polynomial in $x$ and the coefficients of $Q$. We shall establish this by proving the following theorem.

THEOREM 2. *Equation* (1) *holds with*

$$b_{pi} = (-1)^{p-i}\binom{p+1}{i}_{Q,x}$$

*where the quantities* $\binom{p}{i}_{Q,x}$ $(0 \leqq i \leqq p)$ *are polynomials in* $x$ *and the coefficients of* $Q$, *defined by the recurrence*

$$(7)\qquad \binom{p+1}{i}_{Q,x} = \binom{p}{i-1}_{Q,Q(x)} + \binom{p}{i}_{Q,Q(x)} \prod_{\substack{j=0\\ j\neq i}}^{p} Q'(Q^{(i)}(x), Q^{(j)}(x))$$

*where* $Q'(x, y)$ *is the polynomial defined* (*uniquely*!) *by*

$$(x-y)Q'(x,y) = Q(x) - Q(y),$$

*and the initial conditions are*

$$\binom{p}{-1}_{Q,x} = 0 \quad p = 0, 1, \cdots; \quad \binom{0}{i}_{Q,x} = 0 \quad i = -1, 1, 2, \cdots; \quad \binom{0}{0}_{Q,x} = 1.$$

*Explicitly,* $\binom{1}{0}_{Q,x} = \binom{1}{1}_{Q,x} = 1$,

$$\binom{2}{0}_{Q,x} = -b_{10} = \frac{Q^{(2)}(x) - Q(x)}{Q(x) - x} = Q'(x, Q(x))$$

$$\binom{2}{1}_{Q,x} = b_{11} = \frac{Q^{(2)}(x) - x}{Q(x) - x} = 1 + Q'(x, Q(x))$$

$$\binom{3}{0}_{Q,x} = \binom{2}{0}_{Q,Q(x)} \prod_{j=1}^{2} Q'(x, Q^{(j)}(x))$$

$$= Q'(Q(x), Q^{(2)}(x))Q'(x, Q(x))Q'(x, Q^{(2)}(x)).$$

*Proof.* We show by induction that the polynomial recurrence (7) satisfies the formal definition (5). So, suppose this is true for $p = 0, 1, \cdots, n-1$; then for $p = n$ we have to show that

$$\prod_{\substack{j=0\\ j\neq i}}^{p} \frac{Q^{(p+1)} - Q^{(j)}}{Q^{(i)} - Q^{(j)}} = \prod_{\substack{j=0\\ j\neq i-1}}^{p-1} \frac{Q^{(p+1)} - Q^{(j+1)}}{Q^{(i)} - Q^{(j+1)}} - \prod_{\substack{j=0\\ j\neq i}}^{p-1} \frac{Q^{(p+1)} - Q^{(j+1)}}{Q^{(i+1)} - Q^{(j+1)}} \prod_{\substack{j=0\\ j\neq i}}^{p} \frac{Q^{(i+1)} - Q^{(j+1)}}{Q^{(i)} - Q^{(j)}}.$$

But the right-hand side is

$$\frac{1}{\prod_{\substack{k=0\\k\neq i}}^{p}(Q^{(i)}-Q^{(k)})}\left\{(Q^{(i)}-Q^{(0)})\prod_{\substack{k=1\\k\neq i}}^{p}(Q^{(p+1)}-Q^{(k)})\right.$$

$$\left.-(Q^{(i+1)}-Q^{(p+1)})\prod_{\substack{k=1\\k\neq i+1}}^{p}(Q^{(p+1)}-Q^{(k)})\right\}$$

which collapses to equal the left-hand side of (7).

**4. Further generalizations.** In the linear case, $Q'(x, y) = q$, and $\binom{p}{i}_{Q,x} = \binom{p}{i}_q$, independent of $x$; this suggests that it may be productive to search for $(Q, x)$-generalizations of other $q$-formulas. As an example, we consider Theorem 3.3 of Gessel and Stanton [2], called by them a "$q$-analogue of Lagrange inversion for $x/(1 - x)$." As they point out, Gessel and Stanton's result is equivalent to the assertion that the following triangular matrices are mutual inverses:

$$B_{nk}=\frac{(Aq^k)_{n-k}}{(q)_{n-k}}q^{-nk},\qquad 0\leqq k\leqq n$$

$$B_{kl}^{-1}=(-1)^{k-l}\frac{(Aq^l)_{k-l}}{(q)_{k-l}}q^{\binom{k-l+1}{2}+kl}\qquad 0\leqq l\leqq k.$$

First we must write this result in a form involving polynomials in $q$. To avoid negative powers of $q$ we work with

$$B_{nk}^{*}=q^{\binom{n}{2}+\binom{k+1}{2}}B_{nk}=\frac{(Aq^k)_{n-k}}{(q)_{n-k}}q^{\binom{n-k}{2}}$$

$$B_{kl}^{*-1}=q^{-\binom{k+1}{2}-\binom{l}{2}}B_{kl}^{-1}=(-1)^{k-l}\frac{(Aq^l)_{k-l}}{(q)_{k-l}}.$$

The matrix entries are still not polynomials, unless $A = q^a$ for some positive integer $a$; henceforth we assume this. Now we "replace $q^k$ by $Q^{(k)}(x)$," as far as possible. There are many ways of doing this; the simplest we have found gives the following $(Q, x)$-generalization of Gessel and Stanton's result.

THEOREM 3. *The following matrices have polynomial entries and are mutual inverses*:

$$B_{nk}=\frac{(Q^{(a+k)})_{n-k}}{(Q)_{n-k}}q_{n-k}\qquad 0\leqq k\leqq n$$

$$B_{kl}^{-1}=(-1)^{k-l}\frac{(Q^{(a+l)})_{k-l}}{(Q)_{k-l}}\qquad 0\leqq l\leqq k$$

*where*

$$(Q^{(n)})_k=\prod_{j=1}^{k}(x-Q^{(n+j-1)}(x))$$

*and the coefficients $q_k$ are polynomials in $\mathbb{Z}[Q, x]$, and satisfy the recurrence*

$$q_k=\sum_{j=0}^{k-1}(-1)^{k-j-1}\frac{(Q^{(k-j+1)})_j}{(Q)_j}q_j\qquad k=1,2,\cdots.$$

*Proof.* Straightforward substitution shows that the assertion of the theorem is formally correct; we have only to verify that the coefficients $\{q_k\}$ and the matrix entries are indeed polynomials. For this, it is sufficient to prove that the quantities

$$\begin{Bmatrix} a+k \\ k \end{Bmatrix} = \frac{(Q^{(a)})_k}{(Q)_k} = \prod_{j=1}^{k} \frac{(x-Q^{(a+j)})}{(x-Q^{(j)})}$$

are polynomials. But this is evident from the fact that they satisfy the recurrence

$$\begin{Bmatrix} a+k \\ k \end{Bmatrix} = \begin{Bmatrix} a+k-1 \\ k \end{Bmatrix} + \begin{Bmatrix} a+k-1 \\ k-1 \end{Bmatrix} Q^{(a)\prime}(x, Q^{(k)}(x))$$

with the initial conditions

$$\begin{Bmatrix} a \\ -1 \end{Bmatrix} = 0 \qquad \begin{Bmatrix} a \\ 0 \end{Bmatrix} = 1$$

for $a \geqq 0$.

**Acknowledgment.** I thank J. C. Lagarias for helpful comments.

## REFERENCES

[1] G. E. Andrews, *Generalized Frobenius partitions*, Mem. Amer. Math. Soc., 301 (1984), Providence, RI.

[2] I. Gessel and D. Stanton, *Applications of q-Lagrange inversion to basic hypergeometric series*, Trans. Amer. Math. Soc., 277 (1984), pp. 173-201.

[3] J. C. Lagarias and J. A. Reeds, *Unique extrapolation of polynomial recurrences*, SIAM J. Comput., 17 (1988), pp. 342-362.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 371-392, August 1989

# PERFECT GRAPHS AND ORTHOGONALLY CONVEX COVERS*

RAJEEV MOTWANI†, ARVIND RAGHUNATHAN‡§, AND HUZUR SARAN‡¶

**Abstract.** The combinatorial structure of visibility in simple orthogonal polygons is studied. It is shown that the visibility graph of a horizontally or vertically convex polygon is a permutation graph. In general, orthogonal polygons can have concavities (dents) with four possible orientations. In the case where the polygon has three dent orientations, it is shown that the visibility graph is weakly triangulated. Since weakly triangulated graphs are perfect, a polynomial algorithm for this polygon covering problem is obtained. Furthermore, the following duality relationship is obtained. The minimum number of orthogonally convex polygons needed to cover an orthogonal polygon $P$ with at most three dent orientations is equal to the maximum number of points of $P$, no two of which can be contained together in an orthogonally convex covering polygon. Finally, it is shown that in the case of orthogonal polygons with all four dent orientations, the above duality relationship fails to hold.

**Key words.** orthogonal polygons, perfect graphs, minimal coverings, weakly triangulated graphs

**AMS(MOS) subject classifications.** 05, 51, 68

## 1. Introduction.

One of the most well-studied class of problems in computational geometry concerns the notion of *visibility*. Two points in the plane are said to be visible to each other in the presence of obstacles (that are generally polygonal) if there exists a straight-line path between the two points that does not meet any of the obstacles. Other notions of visibility involve paths that are not straight lines, e.g., rectilinear or staircase paths. There is an intimate connection between visibility problems and polygon covering problems. In his recent book on the Art Gallery Problem, O'Rourke [22] states that the fundamental problems involving visibility in computational geometry will not be solved until the combinatorial structure of visibility is more fully understood. In this paper (and a companion paper [21]), we attempt to study this combinatorial structure. A *visibility graph* has vertices that correspond to geometric components, such as points, lines, or regions, and edges that correspond to the visibility of these components to each other. Here, we will be concerned with the visibility graphs for regions inside a simple orthogonal polygon. We show that certain special classes of these visibility graphs are perfect. We use this property of the visibility graphs to devise polynomial algorithms for a class of polygon covering problems that are NP-hard in general.

For our purposes, a *polygon* is a closed, connected set of points in the plane, bounded by several (circular) sequences of straight-line segments. The segments are called edges, their endpoints are called vertices, and their union is called the boundary of the polygon. In our definition, we allow the limiting case where parts of edges of the polygon boundary may coincide. Thus, in the limiting case, we could get "necks" of zero width, and even polygons of zero area. The limiting case can be avoided if polygons are defined as open sets rather than closed sets. The essence of our results is unaffected by the choice of definition, although in the limiting case the details of our proofs differ. The closed version appears more natural and makes our proofs more elegant. A polygon is said to be *simple*

* Received by the editors December 21, 1987; accepted for publication (in revised form) February 1, 1989.

† Computer Science Department, Stanford University, Stanford, California 94305. The work of this author was supported by National Science Foundation grant DCR-8411954. Part of this work was done while the author was at the University of California, Berkeley, California.

‡ Computer Science Division, University of California, Berkeley, California 94720.

§ The research of this author was supported by Semiconductor Research Corporation grant SRC-82-11-008.

¶ The research of this author was supported by National Science Foundation grant CCR-8704184.

if it has no holes, i.e., the polygon boundary is composed of a single sequence of straight-line segments. In this paper, we are only concerned with simple polygons.

An *orthogonal* (or *rectilinear*) polygon (OP), $P$, is a polygon with all its edges parallel to one of the coordinate axes. Let $n$ denote the number of edges on the boundary of $P$. (Note that we are only concerned with simple orthogonal polygons.) An orthogonal polygon is said to be *horizontally convex* (or *vertically convex*) if its intersection with every horizontal (respectively, vertical) line segment is either empty or a single line segment (or a point in the limiting case of a "neck" of zero width). An *orthogonally convex polygon* (OCP) is both horizontally and vertically convex. An *orthogonally star polygon* $Q$ contains a point $p$, such that for every other point $q \in Q$, $p$, and $q$ are contained together in an OCP contained in $Q$. A collection of polygons, $C = \{P_1, P_2, \cdots, P_r\}$ where $P_i \subseteq P$, is said to cover a polygon $P$ if the union of all the polygons in $C$ is $P$. Whenever we speak of a set of covering polygons for an arbitrary polygon $P$, it will be assumed that each covering polygon is totally contained in $P$.

The following classification of orthogonal polygons is due to Culberson and Reckhow [7]. Consider the traversal of the boundary of $P$ in the clockwise direction. At each corner (vertex) of $P$, we either turn 90° right (outside corner) or 90° left (inside corner). Since a 180° vertex is the limit of two 90° vertices as the length of the edge between them goes to zero, we regard a 180° vertex as two outside corners with an edge of length zero between them. Note that in this case, the "neck" of zero width gets traversed in both directions. Thus, when we say a polygon $P$ has $n$ edges, we count the edges of each "neck" twice. A *dent* is an edge of the perimeter of $P$, both of whose endpoints are inside corners. The direction of traversing a dent gives its *orientation*: for instance, a dent traversed from west to east has an N orientation. We will use the natural definition of the compass direction, i.e., the positive direction along the $y$-axis will be referred to as the north direction and so on. Figure 1 illustrates the N, S, E, and W dents. For a dent $D$, $o(D)$ indicates its orientation. Two dents $D_1$ and $D_2$ are said to be similarly oriented if $o(D_1) = o(D_2)$. $D_1$ and $D_2$ are said to be oppositely oriented if $o(D_1) = \mathbf{N}$, $o(D_2) = \mathbf{S}$ (or vice versa), or, if $o(D_1) = \mathbf{E}$, $o(D_2) = \mathbf{W}$ (or vice versa). Otherwise, $D_1$ and $D_2$ are said to have orthogonal orientations. An OP is classified according to the number of orientations of its dents. A class $k$ OP has dents of $k$ different orientations. A class 0 OP does not have dents and is an OCP. A vertically or horizontally convex polygon is a class

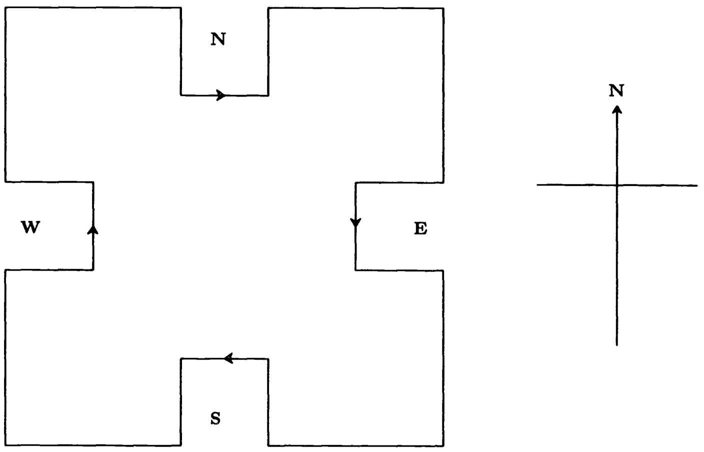


FIG. 1. *Orientation of dents.*

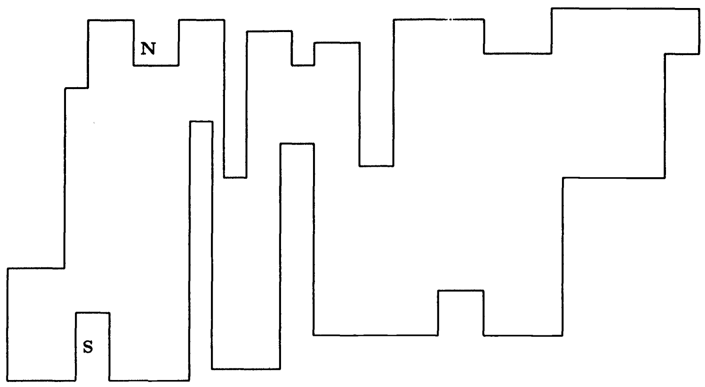


FIG. 2. *A vertically convex polygon.*

2 OP that has only opposing pairs of dents, i.e., either **N** and **S** or **E** and **W** (see Fig. 2). A class 3 OP without N dents is shown in Fig. 3.

The problem of covering general (nonorthogonal) polygons by simpler components has received considerable attention in the literature [5], [6], [17], [18], [29]. It turns out, however, that most of these problems are NP-hard, whether or not the polygon to be covered has holes [1], [8], [29]. The various kinds of orthogonal coverings studied earlier include coverings by rectangles, orthogonally convex polygons, and orthogonally star polygons. Several algorithmic results have been obtained for covering orthogonal polygons. For instance, Franzblau and Kleitman have an $O(n^2)$ algorithm for covering a vertically convex orthogonal polygon without holes with a minimum number of rectangles [10]. Keil has provided an $O(n^2)$ algorithm for covering similar polygons with a minimum number of orthogonally convex polygons [19]. Reckhow and Culberson [24] later provided an $O(n^2)$ algorithm for covering a class 2 orthogonal polygon with a

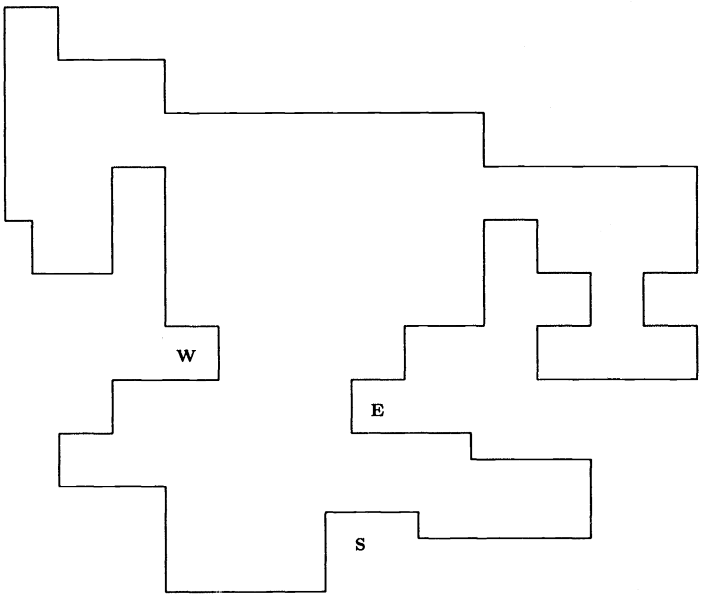


FIG. 3. *A class* 3 *polygon* (*no* N *dents*).

minimum number of orthogonally convex polygons. Recently, Motwani, Raghunathan, and Saran [21] have obtained an $O(n^8)$ algorithm for minimally covering class 4 polygons with orthogonally star polygons, and an $O(n^3)$ algorithm for minimally covering class 3 polygons with orthogonally star polygons.

In this paper, we are more concerned with combinatorial results concerning covering problems for orthogonal polygons. Let an *independent set* of points in a polygon $P$, with respect to a class of covering polygons $C$, denote a set of points in $P$, no two of which can be covered by any covering polygon from the class $C$. A *duality* theorem for covering problems is of the following form. The size of the minimum cover by polygons from class $C$ is equal to the size of the maximum independent set of points with respect to the class $C$. Many interesting duality theorems have been obtained for polygonal covering problems. Chvàtal [10] conjectured that a duality theorem holds for the problem of covering orthogonal polygons by rectangles. This conjecture was shown to be false by Szemeredi and Chung (cited in [4]). However, Chaiken et al. [4] showed that the duality theorem holds for polygons that are orthogonally convex. Gyóri [14] then showed that the duality relationship holds even if the polygon is only vertically (or horizontally) convex. Later, Saks [26] showed that a graph determined by the boundary squares of the grid induced by the vertices of an OCP is perfect. Other related work includes that of Shearer [28], Boucher [3], and Albertson and O'Keefe [2]. The duality theorem has now been shown to hold for covering orthogonal polygons with orthogonally star polygons in [21].

The purpose of this paper is three-fold. First we show that the visibility graph of a vertically convex orthogonal polygon is a permutation graph [12], [20]. Then we show that the visibility graph of a class 3 polygon is a weakly triangulated graph [15]. Furthermore, we prove that a minimum clique cover of the visibility graph corresponds exactly to a minimum cover of an orthogonal polygon by orthogonally convex polygons. Thus, we reduce the polygon covering problem to the problem of covering a weakly triangulated graph with a minimum number of cliques. Since weakly triangulated graphs are perfect [15], we get the following duality relationship for a class 3 polygon $P$: the minimum number of orthogonally convex polygons needed to cover an orthogonal polygon $P$ is equal to the maximum number of points of $P$, no two of which can be contained together in an orthogonally convex covering polygon. Furthermore, the ellipsoid method of Grötschel, Lovász, and Schrijver [13] gives us a polynomial algorithm for the problem of covering a perfect graph with a minimum number of cliques. In practice, however, the ellipsoid method suffers due to severe problems with numerical instability [27]. One reason for this bad performance is that the number of iterations depends on the sizes of the numbers in the input (in other words, it is *not strongly polynomial*). Therefore, the algorithm of [13] can by no means be considered an efficient solution. Hayward, Hoang, and Maffray [16] have obtained an $O(v^5)$ algorithm for the minimum clique cover problem for weakly triangulated graphs, thus providing us with a purely combinatorial algorithm for the polygon covering problem under consideration. Here, $v$ is the number of vertices of the graph. Since this algorithm is combinatorial, it does not suffer from the same drawbacks as the ellipsoid method. The running time for this algorithm has now been improved to $O(v^4)$ [23]. Finally, we show that the visibility graph is not perfect for general (class 4) polygons. Furthermore, we show that the above duality relationship fails to hold for general (or class 4) orthogonal polygons.

We wish to make the point that perfect graphs play a crucial role in polygon covering problems. As pointed out in the previous paragraph, every known instance of the problem of covering orthogonal polygons with a minimum number of OCPs that has a polynomial time solution can be formulated as the minimum clique covering problem of a perfect

TABLE 1
*Classes of polygons and perfect graphs.*

| | | |
|---|---|---|
| Polygon class | OCP | Star |
| | | |
| | | |
| Class 2 | Permutation | Chordal |
| | | |
| Class 3 | Weakly Triangulated | Chordal |
| | | |
| Class 4 | Not Perfect | Weakly Triangulated |
| | | |

graph. By using this graph theoretic technique, Motwani, Raghunathan, and Saran [21] have resolved the problem of covering class 3 and class 4 orthogonal polygons with a minimum number of star polygons by suitably defining a visibility graph that is once again perfect. In fact, the main emphasis of this line of research is to exhibit the relationship between perfect graphs and polygon covering problems and not to provide the most efficient algorithms. Table 1 displays the types of perfect graphs to which the visibility graphs of the various classes of orthogonal polygons belong, for the problems of covering with orthogonally convex and star polygons.

Culberson and Reckhow [7] independently reached the conclusion that a subgraph of the visibility graph of an orthogonally convex polygon with two dent orientations, called the source graph, is a comparability graph [12], [20]. However, permutation graphs are a subset of comparability graphs, and, as such, we believe that our result is stronger. Again, in the case of covering a polygon with three dent orientations, Reckhow [25] has independently shown that the source graph is weakly triangulated [12], [20]. He also provides an $O(n^2)$ geometric algorithm to cover a class 3 polygon with a minimum number of orthogonally convex polygons.

This paper is organized as follows. In § 2, we develop some of the tools required to analyze this problem. Some of the definitions and facts we use are due to Reckhow and Culberson [24]. Section 3 discusses the connection between the covering problem for vertically convex polygons and permutation graphs. In § 4, we state our main results for covering class 3 polygons. Section 5 gives a proof of a technical lemma called the Crossing Lemma that we use to show that the covering problem for class 3 polygons reduces to the clique covering problem for weakly triangulated graphs in §§ 6 and 7, giving us the duality relationship mentioned above. Section 8 shows why we feel that the above techniques will probably not extend to the more general problem of covering a class 4 polygon. We also show that the duality relationship fails to hold for these polygons. In § 9 we will consider possible extensions of our results.

**2. Preliminaries.** In this section we develop some of the tools required to analyze the problem of finding minimum orthogonally convex covers for orthogonal polygons.

Some of the definitions and observations stated here are due to Reckhow and Culberson [24]. Throughout this paper, $P$ refers to the simple orthogonal polygon to be covered.

**2.1. Staircase paths and visibility.** A *maximal* OCP in $P$ is an OCP contained in $P$, but not contained in any other OCP contained in $P$. A *staircase path* in $P$ corresponds to a sequence of points $u = x_0, x_1, \cdots, x_r = v$ contained in $P$ such that (a) each adjacent pair of points, $x_i$ and $x_{i-1}$, determine a vertical or horizontal line segment that is contained in $P$, and (b) in traversing the staircase path from $u$ to $v$, the edges corresponding to the adjacent pairs of points are traversed in at most two of the four possible compass directions. More informally, a staircase path is a connected sequence of horizontal and vertical edges such that the path alternates between left and right turns. We say $u \equiv v$ (read as *u sees v*) if there exists a staircase path joining $u$ and $v$. Note that staircase paths can share points with the boundary of $P$, as $P$ is a closed set. Thus, two points $u$ and $v$ that lie on the same edge of $P$ are visible to each other. We will denote by $s(u, v)$ any fixed staircase path with $u, v \in P$ as its two endpoints. The following two observations [24] demonstrate the inherent relationship between staircase paths and covers by OCPs. Observation 2 is obvious, and requires no proof.

OBSERVATION 1 [24]. *For any two points $u, v \in P$, $u \equiv v$ if and only if some* OCP *(contained in $P$) includes them both.*

*Proof.* Let an OCP, $P'$, include both $u$ and $v$. If $u$ and $v$ lie on a horizontal or vertical line in $P'$, then the line segment between them is contained in $P'$, and hence $u \equiv v$. Otherwise, without loss of generality, let $v$ lie to the northeast of $u$. Construct a path in $P'$ from $u$ to $v$ as follows. Starting at $u$, repeatedly perform the following in order until no further increment to the path is achieved.

(1) Go north until a horizontal edge of $P'$ or the horizontal line through $v$ is reached (whichever comes first).

(2) Go east until a vertical edge of $P'$ or the vertical line through $v$ is reached (whichever comes first).

If we reach $v$, we have a staircase between $u$ and $v$, and $u \equiv v$. Otherwise, let $w$, such that $w \neq v$, be the point reached. We consider two cases. First, let $w$ lie on the same horizontal (respectively, vertical) line as $v$. We then have that points neighboring $w$ along the horizontal to its east (respectively, along the vertical to its north) are not in $P'$. Thus, the horizontal line segment (respectively, vertical line segment), $\overline{vw}$, joins two points contained in $P'$, but is not fully contained in $P'$. This contradicts the definition of an OCP. Second, let $w$ lie neither on the horizontal nor on the vertical through $v$. By construction, $w$ is a point on the boundary of $P'$, such that points neighboring $w$ to its east along the horizontal through $w$, and points neighboring $w$ to its north along the vertical through $w$ are not in $P'$. Also by construction, $v$ is to the northeast of $w$. Now, since $P'$ is connected, some curve $c$ in $P'$ joins $v$ and $w$. Since $v$ is to the northeast of $w$, this curve $c$ will intersect either the vertical through $w$ to its north or the horizontal through $w$ to its east. Thus, there exists some line segment, horizontal or vertical, joining two points in $P'$, and not completely contained in $P'$. This contradicts the fact that $P'$ is an OCP. Thus if an OCP includes $u$ and $v$, we have that $u \equiv v$.

Conversely, if $u \equiv v$, then a staircase path in $P$ joins $u$ and $v$. This staircase path is trivially an OCP that is contained in $P$, as required. □

OBSERVATION 2 [24]. *Any covering of $P$ by* OCP*s can be made into a covering of $P$ by the same number of maximal* OCP*s.*

We say that a staircase path from $u$ to $v$ goes southwest if, in traversing it from $u$ to $v$, we go west on all the horizontal segments and south on all vertical segments. Thus, staircase paths between $u$ and $v$ can be of four possible orientations: northeast, northwest, southeast, southwest. However, depending on the direction of traversal, the same staircase path might be viewed as a northwest/southeast path in one case, or a northeast/southwest path in the other. Thus, we classify staircase paths into two types: In a type I staircase path, we may travel northwest or southeast on it and in a type II staircase path we may travel northeast or southwest on it. A vertical or horizontal staircase path (line) is both of type I and II.

**2.2. Dent lines and zones.** For each dent $D$ of the polygon $P$, we define the notion of a dent line $\vec{D}$ below.

DEFINITION 1. Let $D$ be a dent in an orthogonal polygon $P$. The *dent line* $\vec{D}$ of $D$ is defined to be the *maximal* line segment that is completely contained in $P$ and that contains the dent edge $D$.

Note that under this definition, $\vec{D}$ is constructed by extending $D$ in both directions as long as it is contained in $P$. The orientation of $\vec{D}$ is the same as the orientation of $D$. $\vec{D} - D$ consists of two disjoint line segments $D_l$ and $D_r$, one on each side of $D$. For a dent of S orientation, let $D_l$ be the line segment to the left of $D$ and let $D_r$ be the line segment to the right of $D$ (see Fig. 4). $D_l$ and $D_r$ for dents of other orientations are distinguished by rotating the S dent appropriately.

To simplify stating the following terms and definitions, let $o(D) = \text{S}$. However, it should not be hard to see that similar statements hold for the other three orientations of dents. Consider $P' = P - \vec{D}$. The set of all connected components of $P'$ meeting $\vec{D}$ to their north are collectively termed the $B$ zone, or $B(D)$. The set of all connected components of $P'$ meeting $\vec{D}$ to their south, together with $\vec{D}$ are collectively termed the $A$ zone, or $A(D)$. Since by definition $A(D)$ includes $\vec{D}$, it is a connected polygon. $B(D)$ can be further subdivided into two zones $B_l(D)$ and $B_r(D)$, as follows. The connected components of $B(D)$ meeting $D_l$ are together called $B_l(D)$ and the connected components

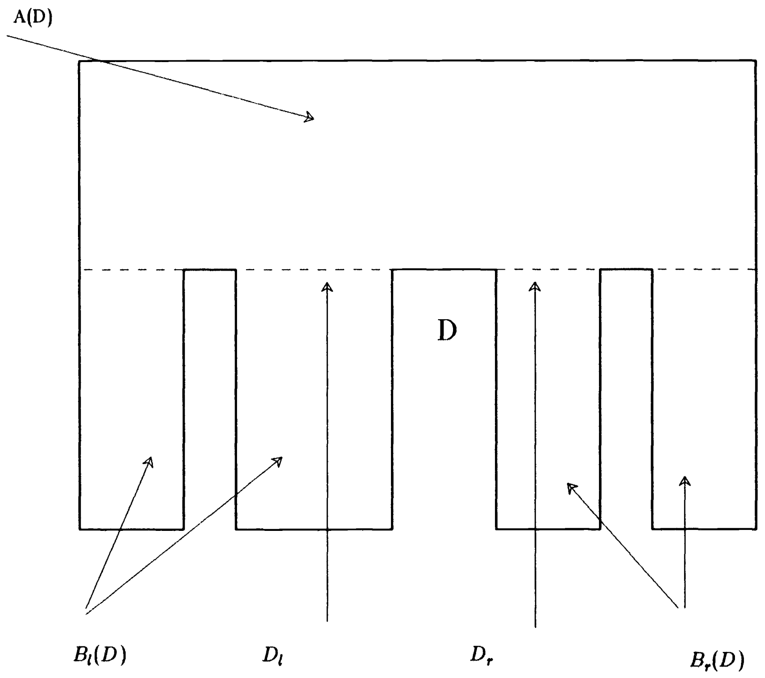


FIG. 4. *Dent lines and zones.*

of $B(D)$ meeting $D_r$ are together called $B_r(D)$. Note that under this definition, no point of $\vec{D}$ is in $B(D)$. Also note that under this definition $B_l(D)$ and $B_r(D)$ need not be connected sets. A $B$ zone is disconnected exactly when two dents with the same orientation share the same dent line. Figure 4 shows how zones are delineated. In our definition of zones, $A(D)$, $P - B_l(D)$, and $P - B_r(D)$ are all three connected subsets of $P$. For any two points $u \in B_l(D)$, $v \in B_r(D)$, there will not exist any staircase path between $u$ and $v$ and, thus, $u \not\equiv v$. We now observe the following facts about dents and zones.

OBSERVATION 3. *Let $u$ and $v$ be two points in $P$. If $u \not\equiv v$ then there exists a dent $D$ such that $u \in B_l(D)$, $v \in B_r(D)$ or $v \in B_l(D)$, $u \in B_r(D)$.*

*Proof.* Assume to the contrary that there exist points $u$ and $v$ in $P$, such that $u \not\equiv v$ and that there exists no dent $D$ in $P$ such that $u \in B_l(D)$, $v \in B_r(D)$ or $v \in B_l(D)$, $u \in B_r(D)$. Now, construct a connected polygon $P'$ from $P$ as follows. For each dent $D$, discard from $P$ the appropriate zone, $B_l(D)$ or $B_r(D)$, such that it contains neither $u$ nor $v$. $P'$ thus defined is guaranteed to contain both $u$ and $v$. $P'$ is also guaranteed to be connected, as at each stage of the construction, we discard only a $B$ zone of a dent (see the paragraph above). Moreover, $P'$ contains no dents. To see this, observe that each time we perform the construction of discarding the appropriate $B$ zone of each dent $D$, exactly one new boundary edge is introduced, and this edge is never a dent of $P'$. Thus, at each stage of the construction, at least one dent is destroyed. Therefore $P'$ is an OCP that includes $u$ and $v$. By Observation 1, we conclude that $u \equiv v$, a contradiction. □

In the case where there is a dent $D$ that satisfies the hypothesis of Observation 3, we say that $D$ *separates* $u$ and $v$, and $D$ itself is called a separating dent for $u$ and $v$. In general, there may be more than one dent separating two points in $P$. In this case, we will focus our attention on any one separating dent.

OBSERVATION 4. *Let $u$, $v$, and $w$ be three points in $P$ such that a dent $D$ separates $u$ from $v$. If $u \equiv w$ and $v \equiv w$, then $w \in A(D)$.*

*Proof.* Dent $D$ separates $u$ from $v$, so we can assume without loss of generality that $u \in B_l(D)$ and $v \in B_r(D)$. Since $u \equiv w$, we have that either $w \in B_l(D)$ or $w \in A(D)$. Again, since $v \equiv w$, we have that either $w \in B_r(D)$ or $w \in A(D)$. Since $B_l(D) \cap B_r(D) = \varnothing$, we have that $w \in A(D)$, as required. □

For a dent $D$, let $L(D)$ denote the infinite line containing $D$. The following observations are stated for particular dent orientations. However, it is not very hard to see that they hold in all of their reflection and rotation symmetric versions.

OBSERVATION 5. *Let $D$ be a dent and $u$, $w$ be points such that $u \in B(D)$ and $w \in A(D)$, and let $u \equiv w$. Without loss of generality, let $o(D) =$ **N**. Then $u$ lies in the half-plane to the north of $L(D)$ and $w$ lies either on $\vec{D}$ or in the half-plane to the south of $L(D)$.*

*Proof.* The boundary between $A(D)$ and $B(D)$ is $\vec{D}$. Any path from a point in $A(D)$ to a point in $B(D)$ must necessarily meet $\vec{D}$, which is of **N** orientation. $u \equiv w$, implying that there is a staircase from $u$ to $w$ in $P$. Any staircase from $B(D)$ to $A(D)$ must cross $\vec{D}$ from north to south. Since $u \in B(D)$ and $w \in A(D)$, we conclude that the staircase from $u$ to $w$ travels southward from $u$. □

OBSERVATION 6. *Let $u$, $v$, and $w$ be points in $P$ such that $u$ lies to the northeast of $w$ and $v$ lies to the northwest of $w$. If $w \equiv u$, $w \equiv v$, and $u \not\equiv v$, then there is a* **N** *dent separating $u$ from $v$.*

*Proof.* If no **N** dent separates $u$ and $v$, then $D$, a dent separating them, must be a **S**, **E**, or **W** dent. If $D$ is an **S** dent, then by Observations 4 and 5, both $u$ and $v$ must be to the south of $\vec{D}$ and $w$ must be either on $\vec{D}$ or to the north of it. Thus, $u$ and $v$ must be to the south of $w$. But we already have that $u$ and $v$ lie to the northeast and northwest of $w$, respectively. If $D$ is a **W** (respectively, **E**) dent, then we can show in a similar fashion that $u$ (respectively, $v$) lies to the west (respectively, east) of $w$, again a contradiction. So $D$ can only be a N dent. $\square$

**2.3. Regions.** The set of all dents of $P$ subdivides $P$ into regions, as defined below. For each point $p$ and each dent $D$ of $P$, we can uniquely specify whether $p$ belongs to $B_l(D)$, $B_r(D)$, or $A(D)$. Let the dents of $P$ be linearly ordered, and let the *zone vector* of a point in $P$ be the correspondingly ordered list of the (unique) zones of each dent to which the point belongs. Let us say that two points $p$ and $q$ of $P$ are related under $R$ if and only if their zone vectors agree. Note that $R$ is an equivalence relation.

DEFINITION 2. The equivalence classes under $R$ are called the *regions* of $P$.

The *zone vector of a region* is the zone vector of a point in that region. It is easy to see that a region is a connected subset of $P$. For if no two dents of the same orientation share the same dent line, then all zones are connected and a region is the intersection of (a finite number of) zones, and is therefore connected. If two dents of the same orientation do share the same dent line, then the connected components of their respective $B$ zones are clearly in different regions, and the same argument applies. It is further clear from the above that the boundary of a region is composed of polygon edges and dent lines. Since there are only $O(n)$ polygon edges and dent lines, and the number of *cells* created in any *arrangement* [9] of $O(n)$ lines in the plane is $O(n^2)$, there are only $O(n^2)$ regions. Note that a region can be a line segment or a point, and thus have zero area under this definition. Figure 5 indicates a polygon with regions that are line segments (marked 9, 10, 11, and 12) and a region that is a point (marked 13). In the following, we relate the notion of visibility by staircase paths with the covering problem.

DEFINITION 3. Let $u'$ and $v'$ be regions in $P$. We say that $u'$ sees $v'$ if and only if some OCP (contained in $P$) includes both $u'$ and $v'$.

OBSERVATION 7 [24]. *Let $u'$ and $v'$ be regions in $P$, and let $u$ and $v$ be arbitrary points in $u'$ and $v'$, respectively. Then, there is a staircase path between $u$ and $v$ if and only if $u'$ sees $v'$.*

*Proof.* If $u'$ sees $v'$, then some OCP (contained in $P$) includes both $u'$ and $v'$. Hence, by Observation 1, there is a staircase path between $u$ and $v$.

Conversely, let there be a staircase path between $u$ and $v$. It is clear from the definition of separation by a dent that there exists no dent $D$ that separates $u$ and $v$. Thus, there

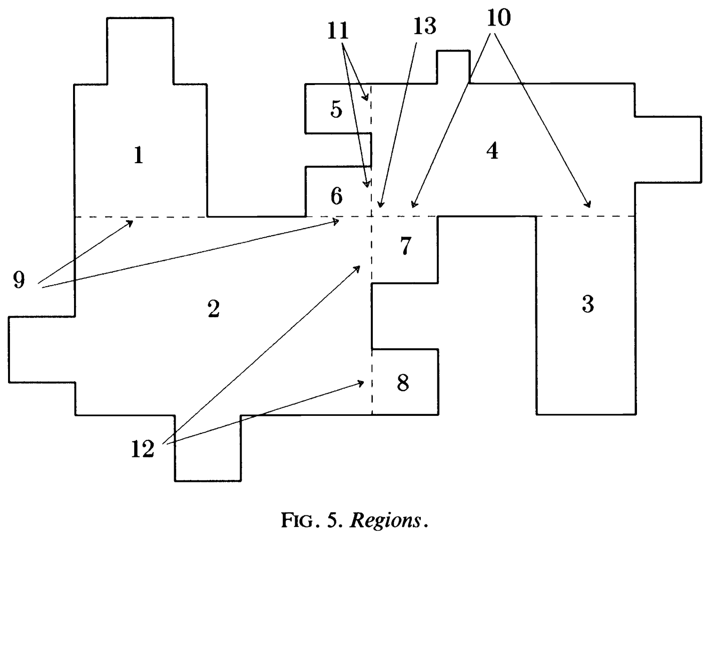


FIG. 5. *Regions.*

exists no dent $D$ such that $u \in B_l(D)$ and $v \in B_r(D)$, or vice versa. This implies that the connected polygon $P'$ obtained from $P$ by discarding the appropriate $B$ zone of each dent, such that neither $u$ nor $v$ is in it, will contain $u$ and $v$. Since the zone vector of every point in a region is identical, both $u'$ and $v'$ are completely contained in $P'$. By our construction above, $P'$ cannot have any dents, and is therefore an OCP by definition. □

**2.4. The visibility graph.** Having defined regions and the notion of visibility between regions, we now study the combinatorial properties of visibility thus defined. The tool we use to study these properties is the *visibility graph* of $P$.

The *visibility graph*, $G(V, E)$, for the polygon $P$, is defined as follows. The vertex set $V$ of $G$ contains a vertex corresponding to each region in $P$. Two vertices $u'$ and $v'$ are adjacent in the graph $G$ if the corresponding regions in $P$ can be covered by a single OCP. We will use the same notation for a region of $P$ and the corresponding vertex in $V$. Thus, we have that $\langle u', v' \rangle \in E$ if and only if $u'$ sees $v'$. It follows from Observation 7 that $\langle u', v' \rangle \in E$ if and only if there is a staircase path from each point in the region $u'$ to each point in the region $v'$. The following lemma provides the relationship between the covering problem for $P$ and the visibility graph $G$.

LEMMA 1. *Let $H(V', E')$ be a complete subgraph (clique) of $G$. Then, the regions of $V'$ can be covered by a single orthogonally convex polygon.*

*Proof.* Let $u', v' \in V'$. Clearly, $u'$ sees $v'$. Thus, for points $u \in u'$ and $v \in v'$, no dent $D$ separates $u$ and $v$. By the definition of regions, we have that there is no dent $D$ such that $u' \subseteq B_l(D)$ and $v' \subseteq B_r(D)$, or vice versa. In other words, if some $w' \in V'$ is in $B_l(D)$ (respectively, $B_r(D)$) for a dent $D$, then every $v' \in V'$ is in $A(D) \cup B_l(D)$ (respectively, $A(D) \cup B_r(D)$). It now follows that for every dent $D$, there exists a $B$ zone, either $B_l(D)$ or $B_r(D)$ that does not contain any of the regions of $V'$.

As in the proof of Observation 7, we can now obtain a connected polygon $P'$ from $P$ by discarding such a $B$ zone for each dent $D$. $P'$ will contain every region of $V'$, and, by construction, has no dents, implying that it is an OCP. □

Since the regions $u'$ and $v'$, such that $u'$ does not see $v'$, cannot both be covered by an orthogonally convex polygon, Lemma 1 implies that a minimum clique cover of $G$ (that is, a minimum cardinality set of cliques of $G$ with every vertex of $G$ belonging to some clique) corresponds exactly to a minimum cover of $P$ by orthogonally convex polygons. Finding a minimum clique cover is NP-hard for general graphs [11], and this formulation of the problem does not give us an algorithm immediately. However, there is an important subclass of graphs (called perfect graphs) for which the minimum clique cover problem can be solved in polynomial time [20]. We will show in a later section that the visibility graph for a class 3 polygon is perfect. Independently, Reckhow [25] has shown that a subgraph of the visibility graph, called the source graph, is perfect. See the Appendix for a definition of source graphs.

**3. Vertically convex polygons and permutation graphs.** In this section, we show that the visibility graph $G$ of a vertically convex polygon $P$ is a permutation graph [12], [20]. A vertically convex polygon is a class 2 polygon with N and S dents only.

A *comparability graph* is one that can be obtained from a partially ordered set $Q$ by taking the elements of $Q$ as its vertices and joining two elements if and only if they are comparable. In other words, it is the undirected version of the transitive closure of $Q$. We now define permutation graphs. Although permutation graphs were originally defined differently, we provide an equivalent definition [20] that is suitable for this paper.

DEFINITION 4. A graph $G$ is a *permutation graph* if both $G$ and its complement are comparability graphs.

Comparability graphs are known to be in the class of graphs called perfect graphs [20]. It follows that permutation graphs are also perfect graphs.

For every region $u'$ of the vertically convex polygon $P$, we pick an arbitrary representative point $u$ and argue about this set of points. Recall that, by Observation 7, regions $u'$ and $v'$ see each other if and only if $u \equiv v$.

The following lemma shows that visibility in a vertically convex polygon is in some sense a transitive property.

LEMMA 2. *Let points $u$, $v$, and $w$ be in $P$, a vertically convex polygon. Let $u \equiv v$, such that the staircase $s(u, v)$ from $u$ to $v$ is horizontal, northwest, or northeast, and let $v \equiv w$, such that the staircase $s(v, w)$ from $v$ to $w$ is horizontal, northwest, or northeast. Then, $u \equiv w$.*

*Proof.* We consider two cases.

*Case* 1. If both staircases are horizontal, clearly $u \equiv w$.

*Case* 2. Let at most one staircase be horizontal. Without loss of generality, let $s(u, v)$ go to the east or northeast from $u$. If $s(v, w)$ goes either north or northeast from $v$, then $u \equiv w$. Assume to the contrary that $u \not\equiv w$. Thus, $s(v, w)$ is west or northwest from $v$. By Observation 6, a **W** dent separates $u$ and $w$. Since $P$ is vertically convex, we obtain a contradiction. □

Given the visibility graph $G$, we construct a directed graph $H_G$ from $G$ as follows. The undirected version of $H_G$ is $G$. Edge $\langle u', v' \rangle$ of $G$ is oriented from $u'$ to $v'$ in $H_G$ (denoted by $u' \twoheadrightarrow v'$) if the representative point $u$ is to the south of the representative point $v$. If two points $u$ and $v$ see each other along a horizontal line, then the edge between them is oriented from west to east. Note that $H_G$ is constructed from $G$ with respect to a particular set of representative points. By our construction of orienting the edges of $G$, $H_G$ is acyclic. To prove that $H_G$ is the transitive closure of some partial order, all we need to show is that if edges $u' \twoheadrightarrow v'$ and $v' \twoheadrightarrow w'$ exist in $H_G$, then the edge $\langle u', w' \rangle$ exists in $G$. But exactly this is shown in Lemma 2: if edges $u' \twoheadrightarrow v'$ and $v' \twoheadrightarrow w'$ exist in $H_G$, then $u$, $v$, and $w$ satisfy the hypothesis of Lemma 2, and hence edge $\langle u', w' \rangle$ does exist in $G$. We have thus shown the following lemma. We note that independently, Culberson and Reckhow [7] have shown that the source graph of a vertically convex polygon is a comparability graph.

LEMMA 3. *The visibility graph $G$ of a vertically convex polygon $P$ is a comparability graph.*

Let us now make the following observation.

OBSERVATION 8. *Let $P$ be vertically convex. Let $u$, $v$, and $w$ be points in $P$ such that $u$ is to the west of $v$, and $v$ is to the west of $w$. Furthermore, let $u \equiv w$. Then, either $v \equiv u$ or $v \equiv w$.*

*Proof.* Since $u \equiv w$, there is a staircase between $u$ and $w$ that goes either northeast or southeast from $u$. Without loss of generality, let it go northeast from $u$. This staircase has a point, $p$ say, either vertically above or below $v$, as $v$ is to the west of $w$ and to the east of $u$. Since $P$ is vertically convex, the line segment $\overline{vp}$ is completely contained in $P$. Let $p$ be vertically above $v$. $v$ has a northeast staircase to $w$ by following the vertical segment from $v$ to $p$ and thence to $w$. Similarly, if $p$ is vertically below $v$, then $v$ has a southwest staircase to $u$. □

We now show that the complement graph $G^c$ is also a comparability graph. As before, we construct a directed graph $H_{G^c}$ from $G^c$, such that the undirected version of $H_{G^c}$ is $G^c$. The orientation of an edge $\langle u', v' \rangle$ is from $u'$ to $v'$ if representative point $u$ is to the west of representative point $v$. (Note that since $P$ is vertically convex, zero area regions can only be horizontal line segments. Hence, we can always pick our set of representative points so that no two such points fall on the same vertical line.) As before,

$H_{G^c}$ is acyclic. Again, as before, we need to show that if $u' \to v'$ and $v' \to w'$ are oriented edges in $H_{G^c}$, then $\langle u', w' \rangle$ is in $G^c$. If $\langle u', w' \rangle$ is not present, then $u \equiv w$. In this case, $u$, $v$, and $w$ satisfy the hypothesis of Observation 8. Thus, we have that either $v \equiv u$ or $v \equiv w$, a contradiction.

The preceding arguments have established the following lemma.

LEMMA 4. *$G^c$, the complement of the visibility graph $G$, is a comparability graph.*

Now, Lemmas 3 and 4 together imply Theorem 1.

THEOREM 1. *The visibility graph $G$ of a vertically convex polygon is a permutation graph.*

## 4. Polygons with three dent orientations.

In this section, we state our main results concerning orthogonal polygons with three dent orientations (class 3 polygons). We first assert that the visibility graph of a class 3 polygon is perfect. In a perfect graph $G$, the size of a minimum clique cover of every induced subgraph $G'$ is equal to the size of a maximum independent set of $G'$. We then state the duality relationship for class 3 polygons. We first need the following definition.

DEFINITION 5. A graph $G$ is *weakly triangulated* if neither $G$ nor $G^c$, the complement of $G$, contain induced cycles of length greater than four.

The following theorem, proved by Hayward [15], will be useful.

THEOREM 2 (Hayward). *Weakly triangulated graphs are perfect.*

We now state our main theorem, the proof of which is contained in the next three sections.

THEOREM 3. *The visibility graph of a class 3 orthogonal polygon $P$ is weakly triangulated.*

Theorem 2, together with Hayward's Theorem, provides us with the following duality relationship.

COROLLARY 1 (The Duality Relationship). *The minimum number of orthogonally convex polygons needed to cover an orthogonal polygon $P$ with at most three dent orientations is equal to the maximum number of points of $P$, no two of which can be contained together in an orthogonally convex covering polygon.*

Since the source graph (see Appendix) is an induced subgraph of the visibility graph, by Theorem 2, it must be perfect. Moreover, the induced subgraphs of a weakly triangulated graph are also weakly triangulated. By Lemmas 1 and 1′ (see the Appendix), we have that a minimum clique cover of the source graph corresponds to a minimum cover of $P$ by maximal OCPs. Hayward, Hoang, and Maffray [16] have devised an $O(v^5)$ algorithm to compute the minimum clique cover of a weakly triangulated graph, where $v$ denotes the number of vertices of the graph. This has recently been improved to an $O(v^4)$ algorithm [23]. This is a significant improvement over the ellipsoid method generally used for perfect graphs. The main advantage of working with source graphs is that, in a class 3 polygon, the number of sources must be $O(n)$ (see the Appendix). It is not very hard to see that a minimal cover by OCPs can be efficiently constructed given the minimum clique cover of $G_s$. Thus, we have an $O(n^4)$ algorithm for the convex cover problem for class 3 polygons. We note that, independently, Reckhow [25] has an $O(n^2)$ algorithm for this problem.

## 5. The crossing lemma.

In this section we prove a technical lemma, called the Crossing Lemma, which will be required in the proof of Theorem 3. Two staircase paths are said to *cross* if they meet at some point. We will only be considering pairs of staircase paths which cross and have distinct endpoints. Observations 9 and 10 are concerned with pairs of crossing staircase paths that are of the same type and of different types, respectively. These observations are valid in all their reflection and rotation symmetric versions.

OBSERVATION 9. *Let points $u_1, v_1 \in P$ be such that $u_1 \equiv v_1$ and $u_1$ lies southwest of $v_1$. Let points $u_2, v_2 \in P$ be such that $u_2 \equiv v_2$ and $u_2$ lies southwest of $v_2$. If staircase paths $s(u_1, v_1)$ and $s(u_2, v_2)$ cross, then $u_2 \equiv v_1$ and $u_1 \equiv v_2$. Moreover, $u_1$ and $u_2$ lie southwest of $v_2$ and $v_1$, respectively.*

*Proof.* $s(u_1, v_1)$ travels southwest from $v_1$ to $u_1$. $s(u_2, v_2)$ travels southwest from $v_2$ to $u_2$ and meets $s(u_1, v_1)$ at some point, say $p$. Thus, we can take $s(u_1, v_1)$ from $v_1$ to $p$ and then take $s(u_2, v_2)$ from $p$ to $u_2$, establishing a southwest staircase from $v_1$ to $u_2$. Similarly, there is a southwest staircase from $v_2$ to $u_1$. □

OBSERVATION 10. *Let points $u_1, v_1 \in P$ be such that $u_1 \equiv v_1$ and $u_1$ lies southwest of $v_1$. Let points $u_2, v_2 \in P$ be such that $u_2 \equiv v_2$ and $u_2$ lies southeast of $v_2$. If $s(u_1, v_1)$ and $s(u_2, v_2)$ cross and $u_1 \not\equiv u_2$, then an* S *dent separates $u_1$ from $u_2$.*

*Proof.* As before, let $s(u_1, v_1)$ meet $s(u_2, v_2)$ at $p$. Now, $p$ sees $u_1$ to its southwest and $u_2$ to its southeast. By Observation 6, there is an S dent separating $u_1$ and $u_2$. □

Let $G'(V', E')$ be a subgraph of the visibility graph $G(V, E)$. For every vertex $v' \in V'$, fix a point (called $v$) that lies in the region of $P$ corresponding to $v'$. For every edge $\langle u', v' \rangle \in E'$, fix a staircase path $s(u, v)$ from point $u$ to point $v$. We call this collection of points and staircase paths an *instantiation* of the subgraph $G'$. Two staircase paths in an instantiation are said to be nonadjacent if the corresponding edges in $G'$ are nonadjacent.

Let $C$ be a $k$-cycle, $k \geqq 4$, in the graph $G$ such that $V(C) = \{v'_0, v'_1, \cdots, v'_{k-1}\}$ and $\langle v'_i, v'_{i+1} \rangle \in E(C)$, for each $i$ (all indices here and in the rest of the paper are modulo $k$).

LEMMA 5 (Crossing Lemma). *If $C$ is an induced cycle of $G$ (i.e., $C$ has no chords) then some pair of nonadjacent staircase paths must cross in some instantiation of $C$ (see Fig. 6). (We can actually prove the stronger result that some pair of nonadjacent staircase paths must cross in <u>every</u> instantiation of $C$. However, the weaker result suffices for our purposes.)*

*Proof.* Consider $C'$, an instantiation of $C$. Assume to the contrary that none of the nonadjacent pairs of staircase paths cross in $C'$. Let $v_0$ be a northernmost point of $C'$.

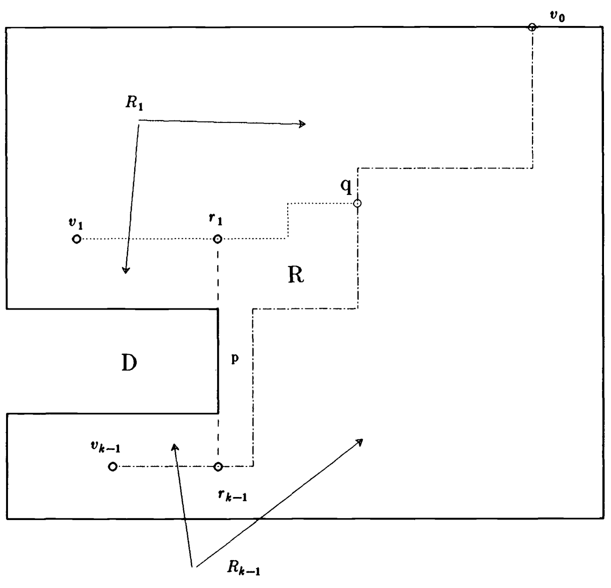


FIG. 6. *Proof of the Crossing Lemma.*

Let $q$ be the last point that $s(v_0, v_1)$ and $s(v_0, v_{k-1})$ have in common, when moving away from $v_0$. Let $s(q, v_1)$ and $s(q, v_{k-1})$ be the appropriate portions of the two staircase paths. Since $v_{k-1} \not\equiv v_1$, a dent $D$ separates $v_1$ and $v_{k-1}$. Without loss of generality, assume that $v_1 \in B_l(D)$ and $v_{k-1} \in B_r(D)$. Since $q$ sees both $v_1$ and $v_{k-1}$, Observation 4 implies that $q \in A(D)$, and that $s(q, v_1)$ and $s(q, v_{k-1})$ intersect $\vec{D}$. Let the intersection points be $r_1$ and $r_{k-1}$, respectively. Let $R$ denote the polygon bounded by the line segment $\overline{r_1 r_{k-1}}$, $s(q, r_1)$, and $s(q, r_{k-1})$, where the staircases $s(q, r_1)$ and $s(q, r_{k-1})$ are defined in the natural way. Thus, $R$ is an OCP.

To simplify the presentation of the proof, we observe that if points $q$ and $v_0$ are not the same, then there is an instantiation of $C$ where $s(v_0, v_1)$ and $s(v_0, v_{k-1})$ are the same between $v_0$ and $q$. This observation is easy to prove and is left to the reader. Let $s(v_0, q)$ denote this common staircase. In the rest of this proof, we assume that $C'$ is an instantiation with the above property.

We now assert that some point, say $v_i \in \{v_2, v_3, \cdots, v_{k-2}\}$ must lie in $R$. Since $R$ is an OCP, this would imply that the vertical line segment going north from $v_i$ to either $s(q, r_1)$ or $s(q, r_{k-1})$ exists in $P$, thus providing a staircase from $v_i$ to $v_0$. This would in turn imply that $v_i \equiv v_0$, a contradiction. The rest of this proof is devoted to proving the above assertion. We will now assume to the contrary that no point of the set $\{v_2, v_3, \cdots, v_{k-2}\}$ is in $R$.

Since $v_0$ is a northernmost point, we can cut $P$ by a horizontal line $l$ passing through $v_0$, and discard the portion of $P$ lying to the north of $l$, to give us $P'$. Let $P'' = P' - R$. $P''$ can in turn be cut by $s(v_0, q)$ to give us two disjoint sets $R_1$ and $R_{k-1}$. $R_1$ is the set that contains $v_1$, and $R_{k-1}$ is the set that contains $v_{k-1}$. It is easy to prove that no point of $R_1$ is in $B_r(D)$ and no point of $R_{k-1}$ is in $B_l(D)$.

We claim that no staircase corresponding to an edge $\langle v'_1, v'_2\rangle, \cdots, \langle v'_{k-2}, v'_{k-1}\rangle$ can intersect $s(v_0, r_1)$, the portion of $s(v_0, v_1)$ between $v_0$ and $r_1$. To see this, note that $v_0$ is the northernmost point in $C'$, and $s(v_0, q)$ is shared by both $s(v_0, v_1)$ and $s(v_0, v_{k-1})$, and by assumption, no two nonadjacent staircases cross. The only staircase that can cross $s(q, r_1)$ is $s(v_1, v_2)$. However, $s(v_1, v_2)$ cannot cross $s(q, r_{k-1})$, as $s(q, r_{k-1})$ is part of $s(v_0, v_{k-1})$ and the corresponding edges are nonadjacent when $k \geqq 4$.

We further assert that no staircase corresponding to an edge $\langle v'_1, v'_2\rangle, \cdots, \langle v'_{k-2}, v'_{k-1}\rangle$ can join a point in $R_1$ and a point in $R_{k-1}$ by crossing $\vec{D}_l$. Assume to the contrary that such a staircase exists. Such a staircase cannot join a point in $R_1$ that is in $A(D)$ to a point in $R_{k-1}$, since $R_{k-1}$ does not contain a point in $B_l(D)$. Thus, it joins a point in $R_1$ that is in $B_l(D)$ to a point in $R_{k-1}$ that is in $A(D)$. Hence, this point cannot be $v_{k-1}$ and the staircase is not $s(v_{k-2}, v_{k-1})$. Moreover, this staircase is now forced to cross $s(q, r_{k-1})$. Since the only staircase that can cross $s(q, r_{k-1})$ is $s(v_{k-2}, v_{k-1})$, we obtain a contradiction.

By the above arguments, we have shown every point of $\{v_2, v_3, \cdots, v_{k-2}\}$ to be in $R_1$. We have further shown that $s(v_{k-2}, v_{k-1})$ cannot leave $R_1$. But, $v_{k-1} \in R_{k-1}$, and this is a contradiction. Therefore, there must be a point $v_i \in \{v_2, v_3, \cdots, v_{k-2}\}$ in $R$, and we have proved the lemma. □

OBSERVATION 11. *Suppose $C$ is an induced $k$-cycle of $G$, where $k \geqq 5$. Let $C'$ be an instantiation of $C$ with two nonadjacent staircases that cross. Any such pair of crossing staircases in $C'$, $s(v_i, v_{i+1})$ and $s(v_j, v_{j+1})$ say, must be of different types.*

*Proof.* Assume to the contrary that $s(v_i, v_{i+1})$ and $s(v_j, v_{j+1})$ cross and are of the same type, say type I. Assume, without loss of generality, that $v_i$ and $v_j$ lie to the southeast of $v_{i+1}$ and $v_{j+1}$, respectively. By Observation 9, we have that $v_i \equiv v_{j+1}$ and $v_j \equiv v_{i+1}$.

Thus, $\langle v'_i, v'_{j+1}\rangle \in E(C)$ and $\langle v'_j, v'_{i+1}\rangle \in E(C)$. Since $k \geqq 5$, one of these edges will cause a chord in $C$ and give us a contradiction. $\square$

**6. Induced cycles in the visibility graph.** In this section we prove the first part of Theorem 3. We show that for a class 3 polygon $P$ the visibility graph $G$ has no induced $k$-cycles, for $k > 4$.

LEMMA 6. *Suppose $P$ is a class 3 polygon (with* **W**, **S**, *and* **E** *dent orientations). Then, the visibility graph, $G$, cannot have induced k-cycles, where $k \geqq 5$.*

*Proof.* Assume to the contrary that $C$ is an induced $k$-cycle in $G$. Let $V(C) = \{v'_0, v'_1, \cdots, v'_{k-1}\}$ denote the set of vertices of $C$ in the cyclic order. By the Crossing Lemma, there exists some instantiation $C'$ of $C$ such that some pair of nonadjacent staircase paths, $s(v_i, v_{i+1})$ and $s(v_j, v_{j+1})$, cross in $C'$. Let us say that they cross at point $q$. By Observation 11, the two staircase paths must be of different types. Assume, without loss of generality, that $s(v_i, v_{i+1})$ is of type I and $s(v_j, v_{j+1})$ is of type II. We may further assume, again without loss of generality, that $v_i$ lies to the southeast of $v_{i+1}$. We do not specify the relative positions of $v_j$ and $v_{j+1}$ right now, but instead say that $v_a$ lies to the southwest of $v_b$, where the sets $\{a, b\}$ and $\{j, j+1\}$ are the same. Since $P$ has only three dent orientations, **W**, **S**, and **E**, we assert that $v_{i+1} \equiv v_b$. If this were not the case, then, by Observation 10, there would be an **N** dent separating the two points, giving us a contradiction. Since the indices of the vertices increase in the cyclic order, and $C$ has no chords, it must be the case that $b = j = i + 2$ and $a = j + 1 = i + 3$. To prevent chords, we have that $v_i \not\equiv v_j$, $v_i \not\equiv v_{j+1}$ and $v_{i+1} \not\equiv v_{j+1}$. By Observation 10, there must be an **E** dent, $D_E$, separating $v_i$ from $v_j$. Similarly, there must be an **S** dent $D_S$, separating $v_i$ from $v_{j+1}$, and a **W** dent $D_W$, separating $v_{i+1}$ from $v_{j+1}$ (see Fig. 7).

By Observation 4, $q \in A(D_W)$ and $q \in A(D_E)$ together imply that $A(D_W) \cap A(D_E) \neq \emptyset$. Therefore, $\vec{D}_E$ must lie to the east of $\vec{D}_W$. We now know that $v_{j+1} \in B_r(D_W)$ and $v_i \in B_l(D_E)$ (see Fig. 8). Moreover, it is clear that $v_i \in A(D_W)$ and $v_{j+1} \in A(D_E)$.

Consider the set of staircase paths that remains after the removal of $s(v_{i+1}, v_j)$ from the instantiation of $C$. Since $v_{j+1} \in B_r(D_W)$ and $v_i \in A(D_W)$, some staircase in the sequence

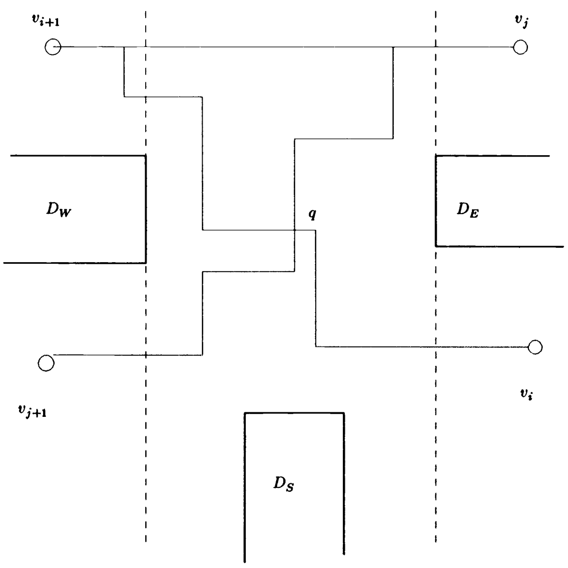


FIG. 7. *Proof of Lemma* 6.

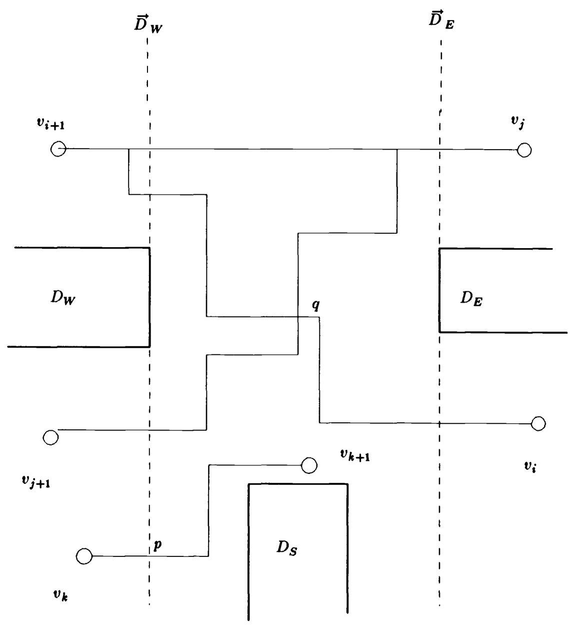


FIG. 8. *Proof of Lemma* 6.

of paths from $v_{j+1}$ to $v_i$, $s(v_k, v_{k+1})$ say, must be the first staircase path to intersect with $\vec{D}_W$. We will assume that $k+1 \neq i$; otherwise, $k \neq j+1$ (since $k \geqq 5$) and we can argue symmetrically about $D_E$. Let the intersection point of $s(v_k, v_{k+1})$ and $\vec{D}_W$ be called $p$ (see Fig. 8). Note that $v_k \in B_r(D_W)$ and $v_{k+1} \in A(D_W)$.

Since $v_{i+1}$ sees $v_i$ to its southeast, and $p$ is to the south of the point of intersection of $\vec{D}_W$ and $s(v_i, v_{i+1})$, there is a northwest staircase path from $p$ to $v_{i+1}$. If the staircase path $s(v_k, v_{k+1})$ is of type I (that is, southeast from $v_k$ to $v_{k+1}$), then $v_{k+1}$ has a northwest staircase path from $v_{k+1}$ to $p$, and thence to $v_{i+1}$. Thus, $v_{k+1} \equiv v_{i+1}$. Since $k+1 \neq i$ and $k+1 \neq j$, we have a chord from $u_{k+1}$ to $v_{i+1}$, a contradiction. If, on the other hand, $s(v_k, v_{k+1})$ is of type II (that is, northeast from $v_k$ to $v_{k+1}$), then again $v_{k+1} \equiv v_{i+1}$; otherwise, $p$ sees $v_{i+1}$ to its northwest and $v_{k+1}$ to its northeast, implying that there must be an N dent separating $v_{k+1}$ from $v_{i+1}$, by Observation 6. Since N dents do not occur in $P$ we again have a chord from $v_{k+1}$ to $v_{i+1}$, a contradiction. □

**7. Induced cycles in the complement of the visibility graph.** In this section, we establish the other part of the proof of Theorem 3, namely, that the complement graph $G^c$ of the visibility graph cannot have any induced cycles of length five or more. The following definitions will prove useful in establishing this result. Let $C$ be a $k$-cycle in the graph $G^c$, where $k \geqq 5$. Let $V(C) = \{v'_0, v'_1, \cdots, v'_{k-1}\}$ denote the vertices of $C$ in a cyclic order. As usual, for each region $v'_i \in V(C)$, we pick an arbitrary representative point (called $v_i$), and work with this set of points. For each $v'_i \in V(C)$ we have $\{v'_{i-1}, v'_{i+1}\} \subseteq N(v'_i, G^c)$, the set of vertices adjacent to $v'_i$ in $G^c$. Recall that $v'_j \in N(v_i, G^c)$ if and only if $v_i \not\equiv v_j$. Hence, $C$ is an induced $k$-cycle if and only if for each $v'_i$ it is the case that $N(v'_i, G^c) \cap V(C) = \{v'_{i-1}, v'_{i+1}\}$.

From Observation 4, we have that if $v_i \not\equiv v_j$ then there must be a dent that separates the two vertices. Let $D_i$ denote the dent that separates the points $v_i$ and $v_{i+1}$ (recall that all indices are modulo $k$). Thus, the $k$-cycle $C$ determines a sequence of $k$ dents corresponding to the $k$ cycle edges in $C$. We first claim that if any two of these $k$ dents are of the same orientation, then there exists a chord for the cycle $C$, unless these two dents correspond to neighboring edges. Assume, without loss of generality, that the three dent orientations in $P$ are $\mathbf{W}$, $\mathbf{S}$, and $\mathbf{E}$.

LEMMA 7. *Suppose the dents $D_i$ and $D_j$ are of the same orientation, where edges $\langle v'_i, v'_{i+1}\rangle$ and $\langle v'_j, v'_{j+1}\rangle$ are nonadjacent; then $C$ cannot be an induced cycle.*

*Proof.* Assume, without loss of generality, that $o(D_i) = o(D_j) = \mathbf{S}$ and the dent line $\vec{D}_i$ lies on the same vertical level or to the north of the dent line $\vec{D}_j$. Note that $D_i$ and $D_j$ could in fact be the same dent. We are assured that the edges $\langle v'_{i+1}, v'_{j+1}\rangle$ and $\langle v'_i, v'_j\rangle$ cannot be present in $G^c$ since they would be chords for the cycle $C$. We are further assured that at least one of $\langle v'_i, v'_{j+1}\rangle$ and $\langle v'_{i+1}, v'_j\rangle$ cannot be present in $G^c$ as $k \geqq 5$.

Since $k \geqq 5$, there must be a vertex in $V(C)$ that is adjacent to neither $v'_i$ nor $v'_{i+1}$ in $G^c$, otherwise there would be chords in $C$. This would imply that there is a vertex that is adjacent to both $v'_i$ and $v'_{i+1}$ in the visibility graph $G$. By Observation 5, we have that both $v_i$ and $v_{i+1}$ must lie to the south of the dent line $\vec{D}_i$. A similar argument also shows that both $v_j$ and $v_{j+1}$ must lie to the south of the dent line $\vec{D}_j$, and thus, also to the south of the dent line $\vec{D}_i$.

We now assert that $v_j \notin A(D_i)$. Suppose $v_j$ did lie in the zone $A(D_i)$. We know that the edge $\langle v'_i, v'_j\rangle$, being absent in $G^c$, is present in $G$, and hence, $v_i \equiv v_j$. We also know that $v_i$ is in one of the $B$ zones of $D_i$. By Observation 5, $v_j$ must be either on $\vec{D}_i$ or in the halfplane to the north of $L(D_i)$ and hence, in the halfplane to the north of $L(D_j)$ also, which is a contradiction. A similar argument shows that $v_{j+1} \notin A(D_i)$ since the edge $\langle v'_{i+1}, v'_{j+1}\rangle$ must be present in the visibility graph $G$, and $v_{j+1}$ lies in the halfplane to the south of $L(D_i)$.

Assume without loss of generality that $v_i \in B_l(D_i)$ and that $v_{i+1} \in B_r(D_i)$. It then follows from the above argument that $v_j \in B_l(D_i)$ and $v_{j+1} \in B_r(D_i)$, since there are staircase paths from $v_i$ to $v_j$ and from $v_{i+1}$ to $v_{j+1}$. Recall that if there is a staircase path between two points inside the polygon, then they cannot lie in different $B$ zones of some dent. Thus, $v_i \not\equiv v_{j+1}$ and $v_j \not\equiv v_{i+1}$, implying that both of $\langle v'_i, v'_{j+1}\rangle$ and $\langle v'_{i+1}, v'_j\rangle$ are edges of $G^c$, a contradiction. □

We are now ready to complete the proof of Theorem 3.

LEMMA 8. *If $G$ is the visibility graph of a class 3 polygon $P$, then $G^c$ cannot have an induced cycle of length five or more.*

*Proof.* Assume to the contrary that $C$ is an induced $k$-cycle in the graph $G^c$. It is clear that $k \geqq 6$ since an induced 5-cycle in $G^c$ would imply the existence of an induced 5-cycle in the visibility graph $G$. From Lemma 7, it follows that dents of the same orientation cannot correspond to two nonadjacent edges of $C$. Thus, we cannot have an induced $k$-cycle where $k \geqq 7$ since only three dent orientations are permitted in $P$. We now complete the proof of the lemma by showing that, given Lemma 7, even induced 6-cycles are not possible.

Consider the case where $k = 6$. From Lemma 7, it follows that we can renumber the vertices of $C$ to ensure that $o(D_0) = o(D_1) = \mathbf{W}$, $o(D_2) = o(D_3) = \mathbf{S}$, and $o(D_4) = o(D_5) = \mathbf{E}$.

Since $C$ is a 6-cycle in $G^c$, there must exist vertices in $C$ that are adjacent (in $G$) to all of $v'_5$, $v'_1$, and $v'_0$, e.g., $v'_3$. Observation 5 then implies that the points $v_5$ and $v_0$ must lie in the halfplane to the east of the $L(D_5)$. Similarly, we can show that the points $v_0$ and $v_1$ must both lie in the halfplane to the west of $L(D_0)$. Thus, $v_0$ is a point that is both to the east of $L(D_5)$ and to the west of $L(D_0)$. This can only happen if $L(D_0)$ lies in the halfplane to the east of $L(D_5)$. Now, since the point $v_3$ has staircase paths to $v_0$, $v_1$, and $v_5$, Observation 5 tells us that it must lie on $\vec{D}_5$ or to the west of $L(D_5)$, and on

$\vec{D}_0$ or to the east of $L(D_0)$. But every point in the halfplane to the west of $L(D_5)$ is in the halfplane to the west of $L(D_0)$ also. Therefore, $v_3$ cannot see all three of $v_0$, $v_1$, and $v_5$. Hence, one of the edges $\langle v'_3, v'_0 \rangle$, $\langle v'_3, v'_1 \rangle$, or $\langle v'_3, v'_5 \rangle$ must be a chord for $C$. This gives us the desired contradiction. □

**8. Orthogonal polygons with four dent orientations.** In this section, we demonstrate arbitrarily large induced odd cycles in the source graph of an orthogonal polygon with four dent orientations. This would show that the source graph, and hence, the visibility graph of a class 4 polygon is not perfect [12], [20], and would also imply that the duality relationship of Corollary 1 does not hold for class 4 polygons.

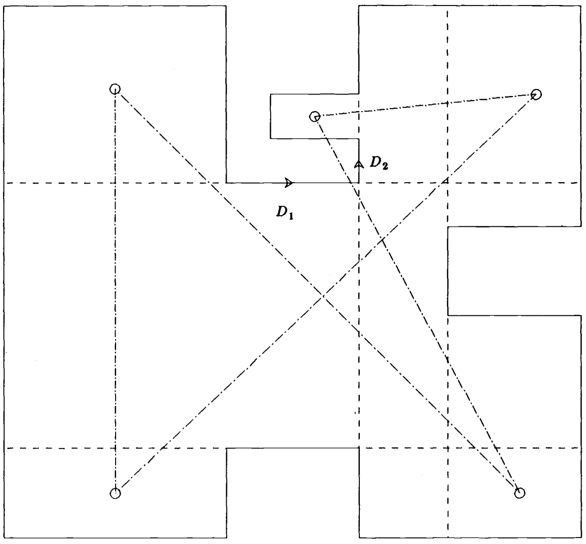


FIG. 9. $P_5$ with induced 5-cycle.

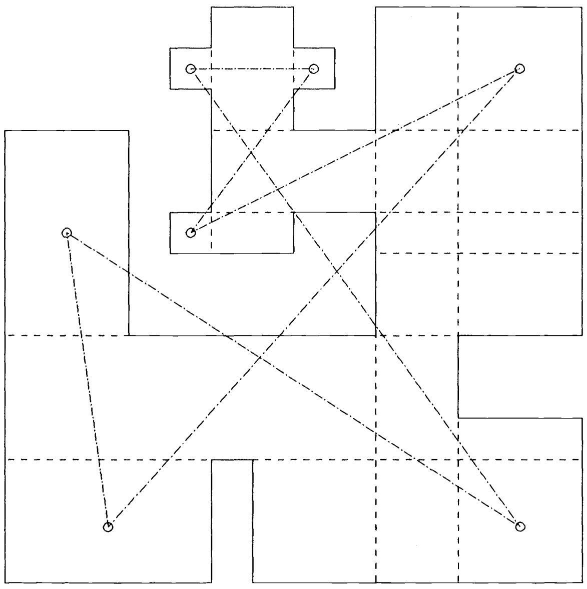


FIG. 10. $P_7$ with induced 7-cycle.

Consider the polygon $P_5$, shown in Fig. 9. There are exactly five sources, but the source graph is a 5-cycle without chords, which is not perfect. Also note that $P_5$ requires three OCPs in any cover, but the size of a maximum independent set in $P_5$ is two. Hence, the duality relationship fails to hold.

Note that $B_l(D_2) \subset B_l(D_1)$. If we now modify $B_l(D_2)$ to obtain the polygon $P_7$, shown in Fig. 10, we find that the source graph is a 7-cycle without chords. A similar construction to $P_7$ would give a polygon whose source graph is a 9-cycle with no chords, and so on to obtain arbitrarily large induced odd cycles.

**9. Further work.** The main contribution of this paper has been the demonstration of the intimate connection between minimum orthogonally convex polygon covers and classes of perfect graphs, and deriving the duality relationship of § 4. The main tool of our analysis has been the visibility graph for regions inside an orthogonal polygon. We have demonstrated certain interesting combinatorial properties of these kinds of graphs. At present, most of the interesting special cases of the problem of covering polygons with simpler polygons that have polynomial time solutions give rise to visibility graphs that are perfect [21], [25], [26], [28]. In the cases of covering simple polygons with convex or star polygons, the visibility graphs are not perfect, and the problems are both NP-hard [1], [8]. Again, in the case of covering orthogonal polygons with a minimum number of rectangles, the visibility graph is not perfect and this problem is also known to be NP-hard [8]. It is our belief that a careful examination of the combinatorial structure of different kinds of visibility graphs may lead to the solution of other open problems in computational geometry.

(1) For the problem of covering class 4 orthogonal polygons with a minimum number of OCPs, the visibility graph is imperfect, as shown in § 8. We conjecture that this problem is also NP-hard.

(2) Is the visibility graph for the problem of covering orthogonal polygons with a minimum number of *rectangular stars* perfect? A rectangular star, or an $r$-star [7] is a polygon that contains a point $q$, such that every other point $p$ of the polygon can be contained together with $q$ in a rectangle (contained in the rectangular star).

**Appendix. Implementation issues: the source graph.** Let us define a *region* DAG (directed acyclic graph) for a simple orthogonal polygon $P$ as follows. This definition is slightly different, though equivalent, to the region DAG of Reckhow and Culberson [24]. We find our viewpoint more convenient to deal with. For each dent $D$, let us say that $B_l(D)$ and $B_r(D)$ are both *dominated by* $A(D)$, while $B_l(D)$ and $B_r(D)$ are *incomparable*. For convenience, we also say that $Z(D)$ dominates itself, where $Z(D)$ is one of $B_l(D)$, $B_r(D)$, or $A(D)$. We say that region $u$ *dominates* region $v$ if every term of the zone vector of $u$ dominates the corresponding term of the zone vector of $v$. In this case, we also say that $v$ *is dominated* by $u$. We say that $u$ and $v$ are *incomparable* if some term of the zone vector of $u$ is incomparable with the corresponding term of the zone vector of $v$. Otherwise, we say that $u$ and $v$ are *comparable*. The region DAG is the underlying partial order of the graph obtained by placing a vertex for each region and adding a directed edge from vertex $v$ to vertex $u$ if region $v$ is dominated by region $u$. Note that $u$ and $v$ are comparable if and only if $u \equiv v$. Otherwise there is a dent $D$ that separates $u$ and $v$, implying that $u$ and $v$ are incomparable.

A *source* is a region of zero in-degree in the region DAG (see Fig. 11). Let us call a dent line $\vec{D}$ of a dent $D$ as *pure* if $\vec{D}$ is not shared by some other dent $D'$, such that $D$ and $D'$ have opposite orientations. Let us further say that two regions $u$ and $v$ are *neighbors* if there is a path from a point in $u$ to a point in $v$ that never leaves $u \cup v$. We now claim that no source is bounded by more than one pure dent line of a given orientation. To

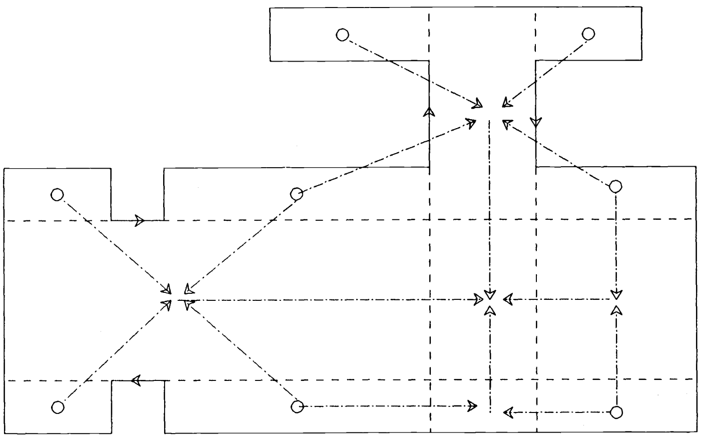


FIG. 11. *Identifying sources.*

see this, we reason as follows. Two pure dent lines $\vec{D}_1$ and $\vec{D}_2$ of the same orientation, S say, are parallel to each other, such that every point on $\vec{D}_1$ (respectively, on $\vec{D}_2$) or neighboring $\vec{D}_1$ (respectively, neighboring $\vec{D}_2$) to its north is in $A(D_1)$ (respectively, in $A(D_2)$). Without loss of generality, let $\vec{D}_1$ be to the south of $\vec{D}_2$. A region $u$ of $P$ that is bounded by both $\vec{D}_1$ and $\vec{D}_2$ has points that are in $A(D_1)$, and in fact, includes points on $\vec{D}_1$ (note that $\vec{D}_1$ is pure). It should be pointed out that points to the north of $\vec{D}_1$ (respectively, $\vec{D}_2$) can belong to a source if $\vec{D}_1$ (respectively, $\vec{D}_2$) is not pure. Let $p \in u$ be a point in $u$. Let $v$ be a region that neighbors $u$ across the common boundary $\vec{D}_1$. Let point $q$ belong to region $v$. We now claim that $u$ dominates $v$, and hence cannot be a source. By our choice of $u$ and $v$, $v$ cannot dominate $u$. If $u$ and $v$ are incomparable, then either some dent separates the two regions, or there exists some dent $D$ such that $u \subseteq B(D)$ and $v \subseteq A(D)$. Both of these are impossible. To see this, note that $u$ and $v$ are neighbors across $\vec{D}_1$, and there is a path joining $p$ and $q$ that only intersects $\vec{D}_1$. Moreover, $\vec{D}_1$ is pure, implying that every dent $D$ that has $\vec{D}_1$ as dent line is of S orientation, and hence $u \subseteq A(D)$ and $v \subseteq B(D)$. Therefore, $u$ dominates $v$, and cannot be a source.

In a class 3 polygon without N dents, every S dent line is pure. It now follows that a source in a class 3 polygon can be bounded by at most one horizontal dent line. It is thus clear that every source in class 2 and class 3 polygons is bounded by some part of the polygon boundary. To count the number of sources of a class 3 polygon, we reason as follows. Let the points where a dent line meets the boundary of $P$ be called *pseudovertices*. Since there are at most $n$ dent lines, there are at most $4 \cdot n$ pseudovertices. There are two kinds of sources in a class 3 polygon: those that are completely bounded by the polygon boundary, and those that are partly bounded by the polygon boundary and partly by dent lines. The first kind of source always includes a vertex of $P$. The second kind of source includes the point where a dent line that is part of its boundary meets a polygon edge that is also part of its boundary. In other words, it includes a pseudovertex. The above argument shows that class 2 and class 3 polygons have only $O(n)$ sources. The following result from [24] shows the importance of sources.

LEMMA 1′ [24]. *If $\beta$ is a set of maximal orthogonally convex polygons that includes every source of $P$, then $\beta$ includes every region of $P$.*

*Proof.* Let $M$ be a maximal OCP that includes a source $u$. Let $v$ be a region such that $u$ is dominated by $v$. From the preceding definitions, we have that every term of the

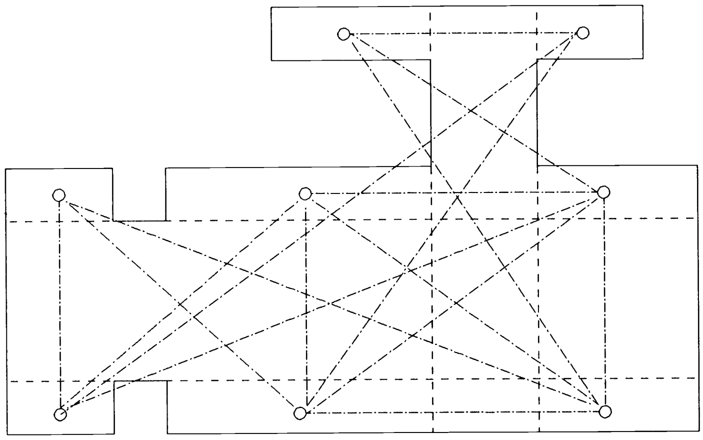


FIG. 12. *The source graph.*

zone vector of $v$ dominates every term of the zone vector of $u$. Let $p$ be an arbitrary point in $M$. It follows from Observation 1 that $p$ sees every point in $u$. Thus, there is no dent $D$ such that $p \in B_l(D)$ and $u \subseteq B_r(D)$, or vice versa. From this and from the domination of $v$ over $u$, we can conclude that if $u \subseteq B_l(D)$ for some dent $D$, then, $v \subseteq A(D) \cup B_l(D)$ and $p \in A(D) \cup B_l(D)$. A symmetric statement can be made about $B_r(D)$. Also, if $u \subseteq A(D)$, then $v \subseteq A(D)$. This shows that there is no dent $D$ that separates $p$ and $v$. Hence, every point in $M$ sees every point in $v$. Therefore, $v$ is contained in $M$, as $M$ is maximal.

To complete the proof of Lemma 1′, we note that every region that is not a source dominates some source. We can, therefore, conclude the hypothesis of the lemma. □

We can now construct the source graph $G_s(V_s, E_s)$ as follows (see Fig. 12). The vertex set $V_s$ has a vertex corresponding to each source region of $P$. As before, we have the edge $\langle u, v \rangle$ in $E_s$ if and only if $u$ sees $v$. Clearly, $V_s \subseteq V$ and the source graph $G_s$ is a vertex induced subgraph of the visibility graph $G$.

Two sources $u$ and $v$, such that $u$ does not see $v$, cannot both be covered by an orthogonally convex polygon. Thus, Lemmas 1 and 1′ together imply that a minimum clique cover of $G_s$ corresponds exactly to a minimum cover of $P$ by maximal orthogonally convex polygons. The advantage of this formulation is that, instead of dealing with $O(n^2)$ regions, we need deal with only $O(n)$ sources for class 2 and class 3 polygons. Furthermore, we can enumerate the sources of a polygon in $O(n^2)$ time [24]. However, implementation issues are not central to the thrust of this paper, and will not be described in any further detail.

Let $u$ and $v$ be sources of $P$. It is easy to see that we can check if $u \equiv v$ by comparing the corresponding terms of their zone vectors and looking for a separating dent. Thus, in $O(n^3)$ time, we can construct $G_s$ for a class 3 polygon.

**Acknowledgments.** The authors are grateful to Gene Lawler for suggesting this field of research. The authors also thank Joe Culberson and Bob Reckhow for sending them accounts of their work, and the two anonymous referees for insightful comments.

## REFERENCES

[1] A. AGGARWAL, *The Art Gallery Theorem: its variations, applications and algorithmic aspects*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, The Johns Hopkins University, Baltimore, MD, 1984.

[2] M. O. ALBERTSON AND C. J. O'KEEFE, *Covering regions with squares*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 240–243.

[3] A. BOUCHER, *It's hard to color antirectangles*, SIAM J. Algebraic Discrete Methods, 5 (1984), pp. 112–163.

[4] S. CHAIKEN, D. J. KLEITMAN, M. SAKS, AND J. SHEARER, *Covering regions by rectangles*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 394–410.

[5] B. CHAZELLE AND D. DOBKIN, *Decomposing a polygon into its convex parts,* in Proc. 11th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1979, pp. 38–48.

[6] B. CHAZELLE, *Computational geometry and convexity,* Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, July 1980.

[7] J. CULBERSON AND R. RECKHOW, *Dent diagrams: a unified approach to polygon covering problems*, Technical Report 87-14, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, July 1987.

[8] ———, *Covering polygons is hard*, in Proc. 29th IEEE Symposium on Foundations of Computer Science, White Plains, NY, October 24–26, 1988.

[9] H. EDELSBRUNNER, *Algorithms in Combinatorial Geometry*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, 1987.

[10] D. S. FRANZBLAU AND D. J. KLEITMAN, *An algorithm for covering regions with rectangles*, Inform. Control, 63 (1984), pp. 164–189.

[11] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, San Francisco, 1979, pp. 53–56.

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

[13] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *The ellipsoid method and its consequences on combinatorial optimization*, Combinatorica, (1981), pp. 169–197.

[14] E. GYÓRI, *A minimax theorem on intervals*, J. Combin. Theory Ser. B, 1 (1984), pp. 1-9.

[15] R. B. HAYWARD, *Weakly triangulated graphs*, J. Combin. Theory Ser. B, 39 (1985), pp. 200–209.

[16] R. B. HAYWARD, C. HOANG, AND R. MAFFRAY, *Optimizing weakly triangulated graphs*, Graphs Combin., to appear.

[17] J. M. KEIL AND J. R. SACK, *Minimum decompositions of polygonal objects*, Computational Geometry, G. T. Toussaint, ed., North Holland, Amsterdam, 1985.

[18] J. M. KEIL, *Decomposing a polygon into simpler components*, SIAM J. Comput., 14 (1985), pp. 799–817.

[19] ———, *Minimally covering a horizontally convex orthogonal polygon*, in Proc. 2nd Annual ACM Symposium on Computational Geometry, Yorktown Heights, New York, Association for Computing Machinery, New York, June 1986, pp. 43–51.

[20] L. LOVÁSZ, *Perfect Graphs*, Selected Topics in Graph Theory 2, Lowell W. Beineke and Robin J. Wilson, eds., Academic Press, London, 1983, pp. 55–85.

[21] R. MOTWANI, A. RAGHUNATHAN, AND H. SARAN, *Covering orthogonal polygons with star polygons: the perfect graph approach*, in Proc. 4th Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, June 6–8, 1988, pp. 211–223.

[22] J. O'ROURKE, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.

[23] A. RAGHUNATHAN, *Algorithms for weakly triangulated graphs*, J. Algorithms, submitted.

[24] R. RECKHOW AND J. CULBERSON, *Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons*, in Proc. 3rd Annual ACM Symposium on Computational Geometry, Association for Computing Machinery, New York, June 1987, pp. 268–277.

[25] R. RECKHOW, *Covering orthogonally convex polygons with three orientations of dents*, Technical Report 87-17, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, August 1987.

[26] M. SAKS, *A class of perfect graphs associated with planar rectilinear regions*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 330–342.

[27] A. SCHRIJVER, *Theory of Linear and Integer Programming*, John Wiley, New York, 1986, pp. 170–171.

[28] JAMES B. SHEARER, *A class of perfect graphs*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 281–284.

[29] J. O'ROURKE AND K. SUPOWIT, *Some* NP-*hard polygon decomposition problems*, IEEE Trans. Inform. Theory, 29 (1983), pp. 181–190.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 393–401, August 1989

# RANDOM SEQUENTIAL ADSORPTION ON GRAPHS*

NICHOLAS PIPPENGER†

**Abstract.** This paper analyzes a process whereby the vertices of a graph are considered in a random sequence, and each considered vertex is "occupied" unless it or an adjacent vertex has previously been occupied. The process continues until no more vertices can be occupied, at which point the "jamming limit" has been reached. The case in which the graph is regular (so that every vertex has degree $d \geqq 2$) and has "few short cycles" is treated. In particular, the results apply to infinite regular trees, to finite graphs obtained from them by forming quotient graphs, and to random regular graphs. It is shown that the probability that a vertex is occupied at the jamming limit tends to $(1 - 1/(d-1)^{2/(d-2)})/2$ as the length of the shortest cycle through it tends to $\infty$. Also treated are graphs that have short cycles but for which every edge is in at most one cycle; in this way approximations are obtained to the occupancy probabilities for two-dimensional triangular, square and hexagonal lattices. Finally, a similar problem is treated in which edges rather than vertices are occupied, and the occupation of an edge prevents the later occupation of edges incident with it. In each case the solution gives the dynamic evolution of the occupancy probabilities, as well as their values at the jamming limit.

**Key words.** packing, monomers, dimers

**AMS(MOS) subject classifications.** 60K35, 82A31, 82A68

## 1. Introduction.

We consider the following random process on a graph. A Poisson stream of "molecules" arrives at each vertex of the graph, the streams arriving at different vertices being independent. When a molecule arrives at a vertex, it "occupies" the vertex, unless that vertex or a vertex adjacent to it has previously been occupied. The process continues until no more vertices can be occupied; this situation is called the "jamming limit." We are interested in determining the density of occupied vertices at the jamming limit. More generally, we are interested in the dynamic evolution of the process as time varies from zero (when no vertices are occupied) to $\infty$ (when the jamming limit is reached). We may take the unit of time to be the mean interarrival time of the Poisson arrival process. Since the various connected components of a graph do not interact in any way during the process, we may assume that the graph is connected.

This process has been studied by chemists and physicists under the name "random sequential adsorption." It is relevant in situations where the reverse process of "desorption" or "evaporation" occurs so slowly that the relaxation time of the system toward equilibrium is long compared with the time of observation. The graphs of interest to chemists and physicists are primarily those modeling polymers and crystal surfaces.

The only graphs for which we know of an exact solution to this problem are those that are essentially "one-dimensional." The simplest of these are those in which each vertex has degree ("coordination number") at most two, in which case the graph must be a path (finite or infinite) or a cycle (finite or infinite). Flory [F] showed that the expected fraction of occupied vertices in a long path or cycle tends to $(1 - e^{-2})/2$ as the length tends to $\infty$. Further results on one-dimensional cases have been presented by Page [Pa], Downton [D], McQuistan and Lichtman [McQL], and McQuistan [McQ].

In this paper we present a method for solving this problem on regular graphs without many short cycles. Among these are the infinite regular trees ("Bethe lattices"), finite regular graphs obtained from them by forming quotient graphs ("Bethe lattices with periodic boundary conditions"), and random regular graphs (for which the number of cycles of any given length remains bounded as the number of vertices tends to $\infty$). We show that the probability that a vertex is occupied at the jamming limit tends to

* Received by the editors May 11, 1988; accepted for publication November 29, 1988.

† Department of Computer Science, University of British Columbia, 6356 Agricultural Road, Vancouver, British Columbia, V6T 1W5, Canada, and IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120.

$(1 - 1/(d-1)^{2/(d-2)})/2$ as the length of the shortest cycle through it tends to $\infty$, where $d > 2$ is the degree. (As $d \rightarrow 2$ we have $1/(d-1)^{2/(d-2)} \rightarrow e^{-2}$, so, in a sense, this expression is correct for $d = 2$ as well.)

We may also consider a process whereby "dimers" arrive at the edges of a graph, and the occupation of an edge prevents the later occupation of an edge incident (sharing a vertex) with it. (For the case $d = 2$, the two processes are equivalent, but for $d > 2$ they differ.) The method of this paper is applicable to this process as well, and shows that the probability that an edge is occupied at the jamming limit tends to $(1 - 1/(d-1)^{d/(d-2)})/d$ as the length of the shortest cycle through it tends to $\infty$. (Again the correct value for $d = 2$ emerges as $d \rightarrow 2$.)

It would of course be of great interest to obtain corresponding results for the triangular, square, and hexagonal ("honeycomb") lattices in two dimensions. While no exact solutions have been reported, Monte Carlo estimates have been made. Widom [W1], [W2] gives $0.38 \pm 0.01$ for the occupancy of the hexagonal lattice; Meakin et al. [Me] give $0.36413 \pm 0.00001$ for the square lattice and $0.23136 \pm 0.00001$ for the triangular lattice.

In the final section of this paper, we compare these Monte Carlo estimates with exact solutions for lattices that are similar to the two-dimensional ones as regards their degree and their girth (that is, the length of their shortest cycle), yet are sufficiently tree-like to allow the methods of this paper to be applied. The simplest such tree-like lattices are of course the infinite regular trees themselves. For degrees 3, 4, and 6, our results give $\frac{3}{8} = 0.375$, $\frac{1}{3} = 0.3333\cdots$ and $(1 - (\frac{1}{5})^{1/2})/2 = 0.27369\cdots$, respectively. While the first of these falls within the error bars of the available Monte Carlo estimate, the other two are disappointing. We can improve these approximations by taking account of the shortest cycles in the two-dimensional lattices, ignoring longer cycles. When this is done, the result for degree 3 and girth 6 is $0.37649\cdots$; for degree 4 and girth 4, $0.35071\cdots$; and for degree 6 and girth 3, $\frac{2}{9} = 0.2222\cdots$. The last two values match the corresponding Monte Carlo estimates much more closely than those obtained by ignoring all cycles.

**2. Random sequential adsorption on graphs.** Let $G = (V, E)$ be an undirected graph with vertices $V$ and edges $E$. Associate with each vertex $v \in V$ a stationary Poisson stream $A(v)$ of independent arrivals, with the streams corresponding to different vertices being independent. We shall assume that each arrival process starts at time zero and that the mean interarrival time for each stream is 1. For the process we are considering, only the first arrival in each stream is important, since only the first molecule to arrive at a vertex has any chance to occupy that vertex. Thus we may turn our attention from $A(v)$ to the *first-arrival time* $t(v)$ for the vertex $v$. Each first-arrival time has an exponential distribution with density $e^{-t}$ on the interval $[0, \infty]$, with the first-arrival times for different vertices being independent.

If $v$ is a vertex in a finite graph $G$ with $n$ of vertices, then there are just $n!$ possible orders of first arrivals. For each of these, $v$ is either occupied or "vacant" (that is, not occupied) in the jamming limit; thus, the probability that $v$ is occupied in the jamming limit is a rational number with denominator dividing $n!$. If $G$ has an automorphism group that acts transitively on the vertices, then this probability is the same for all vertices, and we may speak without ambiguity of the "occupancy probability in the jamming limit."

If $v$ is a vertex in an infinite graph $G$, more must be said. Suppose that all vertices in $G$ have degree at most $d$. Let us say that a sequence $v_0, v_1, \cdots, v_l$ is a "decreasing path" of length $l$ from $v_0$ to $v_l$ if $v_m$ is adjacent to $v_{m-1}$ and $t(v_m) < t(v_{m-1})$ for $1 \leqq m \leqq l$. Let $D_k^{(v)}$ denote the event "there exists a decreasing path of length $k$ from $v$."

LEMMA 2.1.

$$\Pr[D_k^{(v)}] \leq d(d-1)^{k-1}/(k+1)!.$$

*Proof.* There are at most $d(d-1)^{k-1}$ simple paths of length $k$ from $v$, and the probability that such a path is decreasing is $1/(k+1)!$ (since each of the $(k+1)!$ possible orders of arrival is equally likely, and exactly one of them is decreasing). □

Let $E_k^{(v)}$ denote the event "there exists a decreasing path of length $k$ from some vertex that is at distance at most $k$ from $v$."

LEMMA 2.2.

$$\Pr[E_k^{(v)}] \leq d^2(d-1)^{2k-2}/k!.$$

*Proof.* There are at most

$$1+d+d(d-1)+\cdots+d(d-1)^{k-1} \leq (k+1)d(d-1)^{k-1}$$

vertices that are at distance of at most $k$ from $v$, and for each such vertex $w$, the probability that there is a decreasing path of length $k$ from $w$ is as given by Lemma 2.1. □

Suppose that $t(v)$ is the first-arrival time for $v$. Then $v$ will be occupied at $t(v)$ unless some vertex $v_1$ adjacent to $v$ has arrival time $t(v_1) < t(v)$. In the latter case, $v_1$ will be occupied at $t(v_1)$ unless some vertex $v_2$ adjacent to $v_1$ has arrival time $t(v_2) < t(v_1)$. Continuing in this way, we see that whether (and if so, when) the vertex $v$ is occupied depends only on the first-arrival times of vertices on decreasing paths from $v$. In particular, the occupancy of $v$ is well defined unless there is an infinite decreasing path from $v$, and the occupancies of all vertices are well defined unless there is an infinite decreasing path from some vertex in $G$. If there is an infinite decreasing path from some vertex in $G$, then infinitely many of the events $\{E_k^{(v)}\}_{0 \leq k < \infty}$ occur. By Lemma 2.2, the expected number of such events that occur is

$$\sum_{0 \leq k < \infty} \Pr[E_k^{(v)}] \leq d^2 e^{(d-1)^2}/(d-1)^2.$$

By the Borel-Cantelli Lemma, with probability one, only finitely many of these events occur. Thus, with probability one, the occupancy of every vertex in $G$ is well defined.

Let $G_k^{(v)}$ (the "ball of radius $k$ with center $v$") denote the subgraph of $G$ induced by the set of vertices at distance of at most $k$ from $v$. Let $\partial G_k^{(v)}$ (the "sphere of radius $k$ with center $v$") denote the set of vertices in $G_k^{(v)}$ that are adjacent in $G$ to some vertex not in $G_k^{(v)}$. A vertex in $\partial G_k^{(v)}$ is at distance at most $k$ from $v$, since it is in $G_k^{(v)}$. If a vertex in $\partial G_k^{(v)}$ were at distance at most $k-1$ from $v$, then the vertices adjacent to it would be at distance at most $(k-1)+1 = k$ from $v$, and thus would also be in $G_k^{(v)}$. Thus, $\partial G_k^{(v)}$ comprises just those vertices at distance exactly $k$ from $v$ in $G$.

Let $t \in [0, \infty]$ be some fixed time. Let $M^{(v)}$ denote the probability that the vertex $v$ is occupied at time $t$ in $G$, and let $M_k^{(v)}$ denote the probability that the same vertex is occupied at the same time in $G_k^{(v)}$.

LEMMA 2.3.

$$|M^{(v)} - M_k^{(v)}| \leq d(d-1)^{k-1}/k!.$$

*Proof.* The occupancy of $v$ in $G$ can differ from the occupancy of $v$ in $G_k^{(v)}$ only if there is a decreasing path from $v$ to some vertex in $\partial G_k^{(v)}$. If there is such a path, its length must be at least $k$, and thus its initial segment of length $k$ establishes the occurrence of $D_k^{(v)}$. Since the occupancies of $v$ in $G$ and $G_k^{(v)}$ can differ only if $D_k^{(v)}$ occurs, Lemma 2.1 completes the proof. □

In the case of an infinite graph $G$, Lemma 2.3 shows that the occupancy probability of a vertex is approximated by the occupancy probability of that vertex in a sufficiently large but finite neighborhood of the vertex. In particular, it shows that occupancy probabilities in computable graphs (that is, graphs for which the relation of adjacency is effectively computable) are computable real numbers (in the sense that approximations of prescribed accuracy are effectively computable). Lemma 2.3 is significant even for finite graphs; it shows that long-range correlations are weak, and that graphs that are locally similar have similar occupancy probabilities.

**3. Regular graphs with few short cycles.** A regular graph without cycles is an infinite regular tree, or "Bethe lattice." By imposing "periodic boundary conditions" on such a lattice, a finite regular graph with large girth is obtained (see Margulis [Ma] and Imrich [I]; the girth grows logarithmically with the number of vertices). In a regular graph with degree $d$ and girth $g$, balls of radius less than $g/2$ are trees, in which all vertices are either "internal" vertices with degree $d$ or are "leaves" with degree one. We shall determine the occupancy probability for a vertex that is far from the leaves of such a tree; this result will apply to the vertices of regular graphs without short cycles by Lemma 2.3.

A random regular graph with degree $d$ and $n$ vertices (that is, a graph chosen at random with equal probabilities from the set of all such graphs) will probably have some short cycles. A simple calculation shows, however, that the expected number of cycles of length at most $k$ is bounded by a constant depending on $k$ and $d$, but independent of $n$. In a large random regular graph, then, most vertices will not lie on any short cycles, and the results of this section will apply to them by Lemma 2.3. Thus, in a random regular graph, the expected fraction of occupied vertices will be the same as in a regular graph without short cycles, although there may be a small number of exceptional vertices with occupancy probabilities much higher or lower than this average.

Let $d \geqq 2$ be a natural number. We shall begin by determining the occupancy probabilities, as a function of time, for certain vertices in certain trees. Let $A_0$ be a tree containing a single vertex, called its "root." For $k \geqq 1$, if $A_{k-1}$ has been defined, let $A_k$ be the tree obtained from a new vertex, called its "root," together with $d - 1$ disjoint copies of $A_{k-1}$, where the root of $A_k$ is adjacent to the roots of the copies of $A_{k-1}$.

Let $Q_k(t)$ denote the probability that the root of $A_k$ is vacant at time $t$. Clearly, we have $Q_0(t) = 1 - e^{-t}$. The key step of our derivation is the following observation:

$$Q_k(t) = 1 - \int_0^t Q_{k-1}(s)^{d-1} e^{-s}\, ds, \tag{3.1}$$

for $k \geqq 1$. To show (3.1), it suffices to show that the integrand is the rate at which the root is occupied at time $s$. The rate at which first arrivals occur is $e^{-s}$. This must be multiplied by the probability that the $d - 1$ vertices adjacent to the root are vacant at time $s$, which is just $Q_{k-1}(s)^{d-1}$. (The vacancies of the $d - 1$ vertices adjacent to the root are not independent, but if the first arrival at the root occurs at time $s$, then the root is vacant throughout the interval $[0, s)$, and the vacancies of the $d - 1$ vertices adjacent to the root, conditioned on this event, are independent.)

To determine the asymptotics of $Q_k(t)$ as $k \to \infty$, let us consider the fixed point of the transformation $Q_{k-1} \mapsto Q_k$, that is, the solution $Q$ of the integral equation

$$Q(t) = 1 - \int_0^t Q(s)^{d-1} e^{-s}\, ds. \tag{3.2}$$

Differentiation of (3.2) with respect to $t$ yields the differential equation

$$Q'(t) = -Q(t)^{d-1} e^{-t}, \tag{3.3}$$

together with the initial condition $Q(0) = 1$.

The only difficulty in (3.3) comes from the factor $e^{-t}$, which is the density of first arrivals. This difficulty can be removed by the substitution $t = -\ln(1-\tau)$, $\tau = 1 - e^{-t}$, which "uniformizes" the distribution of $\tau$ over the interval $[0, 1]$. Setting

$$P(\tau) = Q(-\ln(1-\tau))$$

for $0 \leqq \tau < 1$, we have

$$P'(\tau) = -P(\tau)^{d-1}, \tag{3.4}$$

together with the initial condition $P(0) = 1$.

Since (3.4) does not involve $\tau$ explicitly, it can be solved by quadratures:

$$\tau = -\int_1^{P(\tau)} \frac{dx}{x^{d-1}}$$
$$= \begin{cases} -\ln P(\tau), & \text{if } d = 2, \\ -(1 - 1/P(\tau)^{d-2})/(d-2), & \text{if } d > 2, \end{cases}$$

where the lower limit of integration is the initial condition $P(0) = 1$. Thus,

$$P(\tau) = \begin{cases} e^{-\tau}, & \text{if } d = 2, \\ 1/((d-2)\tau + 1)^{1/(d-2)}, & \text{if } d > 2. \end{cases} \tag{3.5}$$

This is the desired solution of the transformed version,

$$P(\tau) = 1 - \int_0^\tau P(\sigma)^{d-1}\, d\sigma, \tag{3.6}$$

of (3.2).

To apply this result, let us now define

$$P_k(\tau) = Q_k(-\ln(1-\tau)),$$

for $k \geqq 0$. Clearly, $P_0(\tau) = 1 - \tau$. From (3.1) it follows that

$$P_k(\tau) = 1 - \int_0^\tau P_{k-1}(\sigma)^{d-1}\, d\sigma, \tag{3.7}$$

for $k \geqq 1$. We shall show that $P_k(\tau) \to P(\tau)$ uniformly in $\tau$ as $k \to \infty$.

To this end, let $\Delta_k(\tau) = P_k(\tau) - P(\tau)$. Since $0 \leqq P_0(\tau), P(\tau) \leqq 1$, we have $|\Delta_0(\tau)| \leqq 1$. From (3.6) and (3.7) it follows that

$$\Delta_k(\tau) = \int_0^\tau P(\sigma)^{d-1} - P_{k-1}(\sigma)^{d-1}\, d\sigma$$
$$= -\int_0^\tau \Delta_{k-1}(\sigma)(P(\sigma)^{d-2} + \cdots + P_{k-1}(\sigma)^{d-2}),$$

for $k \geqq 1$. Since $0 \leqq P(\tau), P_{k-1}(\tau) \leqq 1$, we have

$$|\Delta_k(\tau)| \leqq (d-1)\int_0^\tau |\Delta_{k-1}(\sigma)|\, d\sigma.$$

It follows by induction on $k$ that

$$|\Delta_k(\tau)| \leqq (d-1)^k \tau^k / k!. \tag{3.8}$$

Thus $P_k(\tau) \to P(\tau)$ uniformly in $\tau$ as $k \to \infty$.

For $k \geqq 1$, let $A_k^*$ be the tree obtained from a new vertex, called its "root," together with $d$ disjoint copies of $A_{k-1}$, where the root of $A_k^*$ is adjacent to the roots of the copies of $A_{k-1}$. Let $Q_k^*(t)$ denote the probability that the root of $A_k^*$ is vacant at time $t$. By an argument analogous to that used to establish (3.1) we have

$$Q_k^* = 1 - \int_0^t Q_{k-1}(s)^d e^{-s} ds.$$

Setting

$$P_k^*(\tau) = Q_k^*(-\ln(1-\tau)),$$

we obtain

(3.9) $$P_k^*(\tau) = 1 - \int_0^\tau P_{k-1}(\sigma)^d d\sigma.$$

Define

(3.10) $$P^*(\tau) = 1 - \int_0^\tau P(\sigma)^d d\sigma.$$

Then we have the following.

LEMMA 3.1.

$$P^*(\tau) = \begin{cases} (1+e^{-2\tau})/2, & \text{if} d=2, \\ (1+1/((d-2)\tau+1)^{2/(d-2)})/2, & \text{if} d>2. \end{cases}$$

*Proof.* From (3.10) and (3.4) we have

$$\begin{aligned} P^*(\tau) &= 1 - \int_0^\tau P(\sigma)^d d\sigma \\ &= 1 + \int_0^\tau P(\sigma)P'(\sigma)\, d\sigma \\ &= 1 + \int_{P(0)}^{P(\tau)} x\, dx \\ &= 1 + \frac{P(\tau)^2 - P(0)^2}{2}. \end{aligned}$$

The lemma follows by substitution of (3.5). □

LEMMA 3.2.

$$|P_k^*(\tau) - P^*(\tau)| \leqq d(d-1)^{k-1}\tau^k/k!.$$

*Proof.* This follows by an argument analogous to that used to establish (3.8). □

These lemmas give us the limiting value of the vacancy probability for a vertex far from the leaves of a large regular tree. The analysis in this section is summarized by the following theorem.

THEOREM 3.3. *Let $v$ be a vertex in a regular graph $G$ of degree $d \geqq 2$, and suppose that there is no cycle of length at most $2k+1$ through $v$. Then*

$$|Q^{(v)}(t) - Q^*(t)| \leqq 2d(d-1)^{k-1}/k!,$$

*where*

$$Q^*(t) = \begin{cases} (1+e^{-2(1-e^{-t})})/2, & \text{if} d=2, \\ (1+1/((d-2)(1-e^{-t})+1)^{2/(d-2)})/2, & \text{if} d>2. \end{cases}$$

*Proof.* By the hypothesis on $v$ in $G$, there is an isomorphism between $G_k^{(v)}$ and $A_k^*$ that maps $v$ to the root of $A_k^*$. Thus $Q_k^{(v)}(t) = Q_k^*(t)$. The theorem then follows from Lemmas 2.3, 3.1, and 3.2. □

We conclude this section with the remark that a similar analysis applies to the process whereby "dimers" arrive at the edges of a graph, and the occupation of an edge prevents the later occupation of an edge incident with it. In this case, the occupancy probability in the jamming limit is $(1 - 1/(d-1)^{d/(d-2)})/d$.

**4. Regular graphs with nonoverlapping short cycles.** We shall now consider some infinite regular graphs with many short cycles. If all simple cycles have a common length $g$, are uniformly distributed (in the sense that the same number $c$ of simple cycles pass through each vertex), and are nonoverlapping (in the sense that at most one simple cycle passes through each edge), then the methods of the preceding section can be adapted to determine the occupancy probability. (The differential equations that arise in this way will in general not be solvable in closed form, but can easily be integrated numerically.) By varying the degree $d$, the girth $g$, and the parameter $c$, a variety of exactly solvable lattices can be obtained. Our main interest in these lattices is as "approximations" (in some sense that we shall not make precise) to the two-dimensional triangular, square, and hexagonal lattices. The simplest and crudest such approximations are the infinite regular trees of degree 6, 4, and 3. The result of the preceding section gives occupancy probabilities in the jamming limit of $(1 - (\frac{1}{5})^{1/2})/2 = 0.27639\cdots$, $\frac{1}{3} = 0.33333\cdots$, and $\frac{3}{8} = 0.375$, respectively.

Consider the infinite graph obtained by joining triangles so that three triangles meet at every vertex and there are no other simple cycles. This lattice, which corresponds to the choice $d = 6$, $g = 3$, and $c = 3$, may be regarded as the Cayley graph of a free product of three copies of the integers modulo 3; it has the same degree and girth as the triangular lattice, but differs in that every vertex is in three triangles rather than six, and every edge is in one rather than two. The lattice is sufficiently tree-like so that the methods of the preceding section can be adapted to determine the occupancy probability in the jamming limit. The differential equation analogous to (3.4) is

$$P'(\tau) = -P(\tau)^4,$$

with the initial condition $P(0) = 1$. This is the same as for the infinite regular tree with $d = 5$, and the solution is $P(\tau) = 1/(3\tau + 1)^{1/3}$. The occupancy probability in the jamming limit is

$$\int_0^1 P(\tau)^6 \, d\tau = \frac{2}{9}$$
$$= 0.2222\cdots,$$

considerably closer to the Monte Carlo estimate 0.23136 given by Meakin et al. [Me] than the value $0.27639\cdots$ obtained by ignoring all cycles.

Consider now the infinite graph obtained by joining squares so that two squares meet at every vertex and there are no other simple cycles. This lattice, which corresponds to the choice $d = 4$, $g = 4$, and $c = 2$, may be regarded as the Cayley graph of a free product of two copies of the integers modulo 4; it has the same degree and girth as the square lattice, but differs in that every vertex is in two squares rather than four, and every edge is in one rather than two. The differential equation analogous to (3.4) is

$$P'(\tau) = -\frac{2P(\tau)^3+1}{3},$$

with the initial condition $P(0) = 1$. Integrating numerically yields $P(1) = 0.43066\cdots$. The occupancy probability in the jamming limit is

$$\begin{aligned}\int_0^1 (-P'(\tau))^2\,d\tau &= -\int_{P(0)}^{P(1)} \frac{2x^3+1}{3}\,dx \\ &= \frac{3-P(1)^4-2P(1)}{6} \\ &= 0.35071\cdots,\end{aligned}$$

which is considerably closer to the Monte Carlo estimate 0.36413 given by Meakin et al. [Me] than the value $0.33333\cdots$ obtained by ignoring all cycles.

Finally, consider the infinite graph obtained by joining hexagons and edges so that one hexagon and one edge meet at every vertex and there are no other simple cycles. This lattice, which corresponds to the choice $d = 3$, $g = 6$, and $c = 1$, may be regarded as the Cayley graph of a free product of the integers modulo 6 with the integers modulo 2; it has the same degree and girth as $L_3$, but differs in that every vertex is in one hexagon rather than three, and every edge is in zero or one rather than two. There are now two differential equations analogous to (3.4):

$$P_1'(\tau) = -P_2(\tau)$$

and

$$P_2'(\tau) = -\frac{2P_1(\tau)^5+10P_1(\tau)^2+3}{15},$$

with the initial conditions $P_1(0) = P_2(0) = 1$. Differentiating the first with respect to $\tau$, substituting the second, multiplying by the integrating factor $2P_1'(\tau)$, and integrating yields

$$P_1'(\tau)^2 = \frac{2P_1(\tau)^6+20P_1(\tau)^3+18P_1(\tau)+5}{45},$$

where the constant of integration has been chosen to satisfy the initial condition $P_1'(0) = -P_2(0) = -1$. Taking square roots (the initial condition shows that the negative root must be taken) and integrating numerically yields $P_1(1) = 0.30738\cdots$ and $P_1'(1) = -0.49700\cdots$. The occupancy probability in the jamming limit is

$$\begin{aligned}\int_0^1 (-P_1'(\tau))(-P_2'(\tau))\,d\tau &= -\int_{P_2(0)}^{P_2(1)} x\,dx \\ &= \frac{1-P_2(1)^2}{2} \\ &= \frac{1-P_1'(1)^2}{2} \\ &= 0.37649\cdots.\end{aligned}$$

Both this and the value 0.375 obtained by ignoring all cycles are within the error estimate 0.01 for the Monte Carlo value 0.38 given by Widom [W1].

## REFERENCES

[D] F. Downton, *A note on vacancies on a line*, J. Roy. Statist. Soc. Ser. B, 23 (1961), pp. 207–214.

[F] P. J. Flory, *Intramolecular reaction between neighboring substituents of vinyl polymers*, J. Amer. Chem. Soc., 61 (1939), pp. 1518–1521.

[I] W. Imrich, *Explicit constructions of regular graphs without small cycles*, Combinatorica, 4 (1984), pp. 53–59.

[Ma] G. A. Margulis, *Explicit constructions of graphs without short cycles and low density codes*, Combinatorica, 2 (1982), pp. 74–78.

[McQ] R. B. McQuistan, *Vacancy annihilation for one-dimensional dumbbell kinetics*, J. Math. Phys., 10 (1969), pp. 2205–2207.

[McQL] R. B. McQuistan and D. Lichtman, *Exact occupation kinetics for one dimensional arrays of dumbbells*, J. Math. Phys., 9 (1968), pp. 1680–1684.

[Me] P. Meakin, J. L. Cardy, E. Loh, Jr., and D. J. Scalapino, *Maximal coverage in random sequential absorption*, J. Chem. Phys., 86 (1987), pp. 2380–2382.

[Pa] E. S. Page, *The distribution of vacancies on a line*, J. Roy. Statist. Soc., B-21 (1959), pp. 364–374.

[W1] B. Widom, *Random sequential addition of hard spheres to a volume*, J. Chem. Phys., 44 (1966), pp. 3888–3894.

[W2] ———, *Random sequential filling of intervals on a line*, J. Chem. Phys., 58 (1973), pp. 4043–4044.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 402–406, August 1989

# A RAMSEY-TYPE THEOREM FOR ORDERINGS OF A GRAPH*

VOJTECH RÖDL† AND PETER WINKLER†

**Abstract.** It is shown that for any graph $G$ on $n$ vertices, there is a number $N$ (of order at most $n^3(\log n)^2$) and a graph $H$ on $N$ vertices such that for any ordering of the vertices of $G$ and any ordering of the vertices of $H$, there is an order-isomorphism from $G$ into $H$.

**Key words.** Ramsey theory, graph, ordered graph

**AMS(MOS) subject classifications.** 05C55, 05C35, 06F99

Many Ramsey-type theorems have the following flavor. For each fixed object $B$ of size $n$, there is an object $C$ (of size $f(n)$) that is so thoroughly saturated with copies of $B$ that copies of $B$ in $C$ satisfying certain additional conditions can always be found. In the situation considered here, $B$ is an ordered graph $G$ (that is, a graph with linearly ordered vertices); $f(n)$ is roughly of order $n^3$; and $C$ is a graph $H$ all of whose orderings contain a monotone isomorphic copy of $G$.

We in fact obtain a slightly stronger result, where the edges of $G$ are labeled and $H$ is required to work simultaneously for all orderings of $G$.

Considerations of size aside, the existence of such an $H$ has been known for many years.

THEOREM 1. *For every ordered graph $G$ there is a graph $H$ such that every ordering of the vertices of $H$ contains a monotone copy of $G$, that is, a subset whose induced edges and induced order constitute an isomorphic copy of $G$.*

Theorem 1 is a consequence of the following result.

THEOREM 2 [3]. *For every directed acyclic graph $D$ there is a directed acyclic graph $D'$ such that every two-coloring of the arcs of $D'$ contains a monochromatic copy of $D$.*

To get from Theorem 2 to Theorem 1, note that the graph $G$ of Theorem 1 (without its vertex order) may be assumed to be asymmetric, that is, to have no nontrivial automorphism; for otherwise $G$ may be isomorphically embedded in a larger graph with this property. Let $D_1$ be obtained by orienting the edges of $G$ in accordance with the vertex order, and let $D_2$ be a copy of $D_1$ with the directions of the arcs reversed. Let $D$ be the disjoint union of $D_1$ and $D_2$, and choose $D'$ to satisfy Theorem 2. Let $H$ be the undirected mate to $D'$. We associate a 2-coloring of the arcs of $D'$ to an ordering of the vertices of $H$ by coloring an arc "red" if it is oriented along the ordering, and "blue" otherwise. A red copy of $D$ in $D'$ is then the union of a monotone $G$ and an antimonotone $G$ in $H$, and a blue copy likewise.

The existence of $H$ is critical in the proof of a result from [2] having (seemingly) nothing to do with ordering. In any class of finite structures with a notion of embedding, we say that a structure $A$ is *ramsey* if for any $B$ there is a $C$ such that if all the $A$'s in $C$ are 2-colored, then there is a copy of $B$ in $C$ all of whose $A$'s receive the same color.

THEOREM 3 [2]. *In the category of graphs and graph embeddings, a graph is ramsey if and only if it is either complete or independent.*

The case $A = K_2$, $B = K_n$, and $C = K_{r(n)}$ is the classic Ramsey Theorem. However, it is the "only if" part of the theorem whose proof utilizes Theorem 1. Note that the vertices of a graph $A$ that is not complete or independent may be given two orderings

* Received by the editors October 5, 1988; accepted for publication (in revised form) February 1, 1989. This research was supported by Office of Naval Research grant N00014-85-K-0769.

† Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia 30322.

such that the resulting ordered graphs, say $G_1$ and $G_2$, are not isomorphic. Choose $H_1$ and $H_2$ as in Theorem 1, so that any ordering of $H_i$ contains a monotone copy of $G_i$. Let $B$ be the disjoint union of $H_1$ and $H_2$. Then in any graph $C$, we may order the vertices arbitrarily and color a copy of $A$ red when its induced order is isomorphic to $G_1$, and blue otherwise. But then every copy of $B$ in $C$ will contain both red and blue copies of $A$.

The proof of Theorem 2 in [3] produces a $D'$ that is superexponential in size relative to the size of $D$. Thus it is perhaps surprising that an $H$ of Theorem 1 can be found whose size is only polynomial in the size of $G$. In fact, a technique used in [1] and [4] for similar problems can be used to obtain an $H$ with roughly $n^4$ vertices, where $n$ is the number of vertices of $G$.

The technique used below employs the projective planes of [1] and [4] but in a somewhat more sophisticated manner, "blowing up" points to a certain size. The result is a bound of $n^3 \log^2 n$; we also will show that the exponent "3" is, in our formulation, the best possible.

DEFINITION. Let $f(n)$ be the smallest integer $N$ satisfying the following. Let

$$c: E(K_n) \to \left\{1, 2, \cdots, \binom{n}{2}\right\}$$

be a labeling of the edges of $K_n$; then there exists a labeling

$$C: E(K_N) \to \left\{1, 2, \cdots, \binom{n}{2}\right\}$$

of the edges of $K_N$ such that for every ordering $<$ of the vertices $V(K_N)$, and every ordering $<'$ of the vertices $V(K_n)$, there is a monotone isomorphic embedding from $K_n$ to $K_N$, i.e., a 1-1 mapping $\phi$: $V(K_n) \to V(K_N)$ satisfying the following:

(i) $\phi(u) < \phi(v)$ for $u <' v$; and

(ii) $C(\{\phi(x), \phi(y)\}) = c(\{x, y\})$ for every $\{x, y\} \in E(K_n)$.

THEOREM 4. *There are constants $c_1$ and $c_2$ such that $c_1 n^3 \leqq f(n) \leqq c_2 n^3 \log^2 n$.*

*Proof.* We begin by establishing the upper bound.

Let $m - 1$ be the nearest prime power to $5n \log n$ and let $P$ be a projective plane of order $m - 1$. Each point $p$ of $P$ is now "blown up" $n$ times to obtain a set $B(p)$. To each line $L$ of the $P$ we then assign a blown-up line $B(L) = \cup\{B(p): p \in L\}$, so that $|B(L)| = mn \approx 5n^2 \log n$.

Let $B(P)$ be the set of all points of the blown-up projective plane $P$, i.e., $B(P) = \cup\{B(p): p \in P\}$. Then $|B(P)| = N$, where $N = n(m^2 - m + 1) \approx 25n^3 \log^2 n$. Let $K_N$ be the complete graph on vertex set $B(P)$, and let $K_n$ be the complete graph on vertices $1, 2, \cdots, n$. Our aim is to color pairs $\{u, v\}$, $u \in B(p)$, $v \in B(q)$, $p \neq q$, randomly by $\binom{n}{2}$ colors in such a way that with large probability (i) and (ii) will be satisfied.

Let us denote by $G$ a complete $n$-partite graph with partition classes $V_1, V_2, \cdots, V_n$, each of size $m$. Let

$$\tilde{c}: E(G) = \left[\bigcup_{k=1}^{n} V_i\right]^2 \to \left\{1, 2, \cdots, \binom{n}{2}\right\}$$

be a coloring of the edges of $G$ that satisfies $\tilde{c}(v_i, v_j) = c(i, j)$ for every $i, j$ and $v_i \in V_i$, $v_j \in V_j$.

For each "blown up" line $B(L)$, let

$$\hat{\Psi}_{B(L)}: V(G) \to B(L)$$

be a random mapping of $G$ onto the points of $B(L)$, where $\hat{\Psi}_{B(L)}$ and $\hat{\Psi}_{B(L')}$ are chosen independently for $L \neq L'$. We are now ready to define a random mapping $\hat{C}$ that will, with probability approaching one, satisfy our requirements. For $u \in B(p)$, $v \in B(q)$, $p \neq q$, set

$$\hat{C}(u,v) = \tilde{c}(\hat{\Psi}_{B(L)}^{-1}(u), \hat{\Psi}_{B(L)}^{-1}(v))$$

where $L$ is the line determined by $p$ and $q$.

For now, we fix orderings $<$ of $B(P)$ and $<'$ of $V(K_n)$. We wish to estimate the probability that there is no order- and color-preserving isomorphism from $K_n$ into $K_N$; we begin by bounding the probability that there is no such isomorphism inside $B(L)$ for fixed $L$.

To that end we call a bijection $\Psi_{B(L)}$: $V(G) \twoheadrightarrow B(L)$ *bad* if there is no order-preserving map $\lambda$: $V(K_n) \twoheadrightarrow B(L)$ satisfying $c(i,j) = C(\lambda(i), \lambda(j))$. (Here $C$ is determined by $\Psi_{B(L)}$ in the same way that $\hat{C}$ is determined by $\hat{\Psi}_{B(L)}$.)

Without loss of generality, we now assume that $1 <' 2 <' \cdots <' n$; let $\Psi_{B(L)}$ be a bad bijection. We define a sequence of points $\{a_i\}_{i<n}$ as follows.

Let $a_1$ be the first point (with respect to $<$) of $\Psi_{B(L)}(V_1)$, and let $p_1$ be defined by $a_1 \in B(p_1)$.

Suppose $a_1, \cdots, a_i$ and $p_1, \cdots, p_i$ are defined; let $a_{i+1}$ be the $<$-first point of

$$\left(\Psi_{B(L)}(V_{i+1}) - \bigcup_{j=1}^{i} B(p_i)\right) \cap \{a : a > a_i\}$$

and let $p_{i+1}$ be such that $a_{i+1} \in B(p_{i+1})$. If $a_n$ exists, then $\lambda: i \twoheadrightarrow a_i$ is a mapping satisfying $c(i, j) = C(\lambda(i), \lambda(j))$, thus contradicting our choice of $\Psi_{B(L)}$. Let $k$ be the smallest number for which $a_k$ fails to exist; then define $a_0$ to be the $<$-first point, and $a_k$ the $<$-last point, of $B(L)$.

We then have that each bad bijection gives rise to a sequence of $k \leqq n$ *intervals* $[a_{i-1}, a_i)$, $i = 1, 2, \cdots, k$, with the property that

$$[a_{i-1}, a_i) \cap \left(\Psi_{B(L)}(V_i) - \bigcup_{j=1}^{i-1} B(p_j)\right) = \varnothing.$$

For each $i = 1, 2, \cdots, k$, set

$$B_i = [a_{i-1}, a_i) - \bigcup_{j=1}^{k-1} B(p_j).$$

Then the following observations can be made:

(1) The set $B_i$ does not contain any part of $\Psi_{B(L)}(V_i)$ and the set $B(L)$ decomposes as shown in Fig. 1;

(2) $|\bigcup_{i=1}^{k-1} B(p_i)| = n(k-1) < n^2$;

(3) Thus, $\sum_{i=1}^{k} |B_i| \geqq n(m-n)$;

(4) There are fewer than

$$\binom{mn}{1} + \binom{mn}{2} + \cdots + \binom{mn}{n-1}$$

systems of points $a_1 < a_2 < \cdots < a_{k-1}$, $k \leqq n$.

Combining (1)–(4) and using the fact that each of the sets $V_i$ can be permuted in $m!$ ways, we infer that the number of bad bijections $\Psi_{B(L)}$ is bounded from above by

$$\binom{mn}{n}(m!)^n(n-1)^{n(m-n)}n^{n^2}.$$

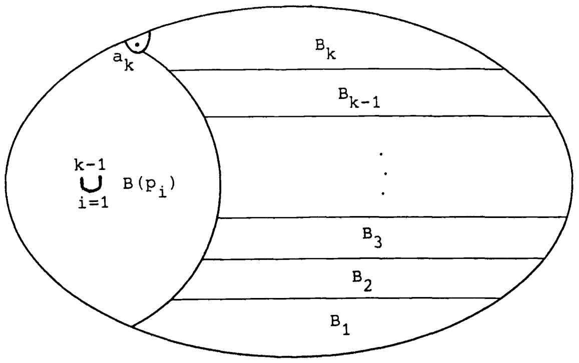


FIG. 1

On the other hand, the total number of bijections $\Psi_{B(L)}$: $V(G) \to B(L)$ is equal to $(mn)! > (mn/e)^{mn}$. Thus for a *random* bijection $\hat{\Psi}_{B(L)}$, we have

$$(*) \qquad \text{Prob}\,(\Psi_{B(L)} \text{ is bad}) < \frac{(m!)^n (n-1)^{n(m-n)} n^{n^2}}{(mn/e)^{mn}} \binom{mn}{n}.$$

Stirling's formula implies that $m! < (m/e)^m \sqrt{7m}$ and thus the left-hand side of $(*)$ can be further bounded from above by

$$\binom{mn}{n} \frac{(m/e)^{mn}(n-1)^{n(m-n)} n^{n^2} e^{mn} (7m)^{n/2}}{(mn)^{n(m-n)}(mn)^{n^2}} < \left(1-\frac{1}{n}\right)^{n(m-n)} (7m)^{n/2}(me)^n$$
$$< e^{n-m}(60m^3)^{n/2}.$$

On the other hand, there are $N/n$ lines and the mappings $\Phi_{B(L)}$ are independent of one another. We can thus conclude that if $E$ is the event that there is *no* map $\lambda$: $V(K_n) \to V(K_N)$ such that $c(i, j) = \hat{C}(\lambda(i), \lambda(j))$ and $\lambda(i) < \lambda(j)$ whenever $i <' j$, then

$$\text{Prob}\,(E) < [(60m^3)^{n/2} e^{n-m}]^{N/n}.$$

There are $N!$ orderings $<$ of $B(P)$ and $n!$ orderings $<'$ of $V(K_n)$, and thus if

$$P = \text{Prob}\,(\text{orderings} <, <' \text{exist for which } E \text{ holds}),$$

then

$$P < N! n^n [(60m^3)^{n/2} e^{n-m}]^{N/n} < [N^n (60m^3)^{n/2} e^{n-m}]^{N/n}.$$

It follows that

$$\frac{\log P}{N/n} < n \log N + \frac{n}{2} \log (60m^3) + n - m$$

$$\approx 3n \log n + n \log (25 \log^2 n) + \frac{3}{2} n \log n + \frac{n}{2} \log (60 \log^3 n) + n - 5n \log n$$

$$\approx \left(3 + \frac{3}{2} - 5\right) n \log n \to -\infty$$

so that $P \to 0$. It follows that *most* collections of mappings $\Psi_{B(L)}$ yield a graph with the desired properties; since the cardinality of this graph is $N \approx 25n^3 \log^2 n$, the upper bound in Theorem 4 is established.

To prove that $f(n) > c_1 n^3$, it suffices to show that $f(n) > f(n-2) + 6c_1 n^2$ for sufficiently large $n$. We do this for $c_1 = \frac{1}{15}$.

Fix a large number $n$ and a bijective edge-labeling $c: E(K_n) \to \{1, 2, \cdots, \binom{n}{2}\}$. Set $N = f(n)$ and let $C$ be a labeling of the edges of $K_N$ by the same label set such that for any ordering $<$ of $V(K_N)$ and any ordering $<'$ of $V(K_n)$, there is a color- and order-preserving isomorphism of $K_n$ into $K_N$.

Let $j$ be the least-represented color in $K_N$, so that

$$|\{e \in E(K_N): C(e) = j\}| \leqq \binom{N}{2} \Big/ \binom{n}{2} < \frac{N^2}{n^2}.$$

We claim that $K_N$ contains a set $X$ of at least $2n^2/5$ vertices such that no two vertices in $X$ are connected by an edge of color $j$. For, let $H$ be the graph on vertices $\{1, 2, \cdots, N\}$ whose edges are exactly the $j$-colored edges of our $K_N$; then by Turán's Theorem [5], if $H$ has no independent set of size $\leqq .4n^2$ then the edges of $H$ must number at least

$$(.4n^2 - 1)\binom{\lfloor N/(.4n^2-1)\rfloor}{2} \approx \frac{1.25N^2}{n^2},$$

contradicting the above bound.

We now restrict the ordering $<'$ of $V(K_n)$ so that the first two vertices are $x$ and $y$, where $c(\{x, y\}) = j$. Next we specify that the elements of $X$ precede all other vertices of $K_N$ in the ordering $<$.

As a result any color- and order-preserving isomorphism $\lambda$ of $K_n$ into $K_N$ sends at most one vertex of $K_n$ into $X$; thus, by extending any ordering of $V(K_n) - \{x, y\}$ and any ordering of $V(K_N) - X$, we may use $\lambda$ to embed $K_{n-2}$ into $K_{N-.4n^2}$ with appropriately restricted coloring $C$.

It follows that $f(n-2) \leqq f(n) - .4n^2$, and the proof of Theorem 4 is complete.

## REFERENCES

[1] J. BROWN AND V. RÖDL, *A Ramsey-type problem concerning vertex partition*, preprint.

[2] J. NEŠETŘIL AND V. RÖDL, *Partition of subgraphs*, in Recent Advances in Graph Theory, M. Fiedler, ed., Academia, Prague, 1985, pp. 413–423.

[3] V. RÖDL, *A generalization of Ramsey's Theorem and the dimension of a graph*, Ph.D. thesis, Charles University, 1973. (In Czech.) *A generalization of the Ramsey Theorem in graphs, hypergraphs and block systems*, Zielona Gora, 1976, pp. 211–219. (In English.)

[4] ———, *Selective graphs*, in preparation.

[5] P. TURÁN, *On an extremal problem in graph theory*, Mat. Fiz. Lapok, 48 (1941), pp. 436–452.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 407–412, August 1989

# NODE-PACKING PROBLEMS WITH INTEGER ROUNDING PROPERTIES*

S. K. TIPNIS† AND L. E. TROTTER, JR.‡

**Abstract.** This paper considers an integer programming formulation of the node-packing problem $\max\{1\cdot x: Ax \leqq w,\ x \geqq 0,\ x \text{ integral}\}$, and its linear programming relaxation, $\max\{1\cdot x: Ax \leqq w,\ x \geqq 0\}$, where $A$ is the edge-node incidence matrix of a graph $G$ and $w$ is a nonnegative integral vector. An excluded subgraph characterization quantifying the difference between the values of these two programs is given. One consequence of this characterization is an explicit description for the "integer rounding" case. Specifically, a characterization is given for graphs $G$ with the property that for every subgraph of $G$ and for any choice of $w$ the optimum objective function values of these two problems differ by less than unity.

**Key words.** node-packing, integer rounding, optimization gap

**AMS(MOS) subject classifications.** 05C70, 68C75, 90C10, 90C35

## 1. Introduction.

Let $G = (V, E)$ be a graph with node set $V$ and edge set $E$. We restrict attention throughout to finite, undirected, loopless graphs. An *independent* (*stable*) set of nodes is a subset $S \subseteq V$ such that no two nodes of $S$ are joined by an edge of $G$. Let $\alpha(G)$ denote the size of a largest stable set in $G$. A subset $F \subseteq E$ is an *edge-cover* if every node of $G$ is the endpoint of some edge in $F$. Denoting $\delta(G)$ as the size of a smallest edge-cover in $G$, $G$ is called a *König graph* when $\alpha(G) = \delta(G)$. A well-known theorem of König states that bipartite graphs without isolated nodes are König graphs. It is not difficult, however, to see that the converse is false, even for graphs without isolated nodes. Characterizations of König graphs are given in Deming [1979], Sterboul [1979], Korach [1982], and Lovász and Plummer [1986]; the latter reference contains a thorough discussion of the subject.

We model the problem of determining $\alpha(G)$ as follows. The *edge-node incidence matrix* of G is the $|E| \times |V|$ matrix $A = [a_{ev}]$ given by

$$a_{ev} = \begin{cases} 1, & \text{if edge } e \text{ is incident to node } v; \\ 0, & \text{otherwise.} \end{cases}$$

For $w$ a nonnegative, integral vector whose components correspond to the edges of $G$, we consider

$$\alpha_w(G) \equiv \max\{1\cdot x: Ax \leqq w, x \geqq 0, x \text{ integral}\},$$

with 1 denoting an appropriately dimensioned vector of ones. We denote the corresponding linear programming relaxation by

$$\bar{\alpha}_w(G) \equiv \max\{1\cdot x: Ax \leqq w, x \geqq 0\}.$$

Then the quantities $\alpha_w(G)$ and $\bar{\alpha}_w(G)$ are finite if and only if $G$ has no isolated nodes. When $\alpha_w(G)$ is finite, we have $\alpha(G) = \alpha_1(G)$ and

$$\delta(G) = \min\{1\cdot y: yA \geqq 1, y \geqq 0, y \text{ integral}\}.$$

* Received by the editors December 17, 1987; accepted for publication (in revised form) January 5, 1989. This work was partially supported by National Science Foundation grant ECS-8504077 and by an Alexander von Humboldt Foundation U.S. Senior Scientist Award.

† Department of Mathematics, Illinois State University, Normal, Illinois 61761.

‡ School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853.

A *subdivision* of $G$ is obtained by replacing the edges of $G$ by simple paths, i.e., by inserting new nodes of degree two into the edges. An *even subdivision* results when the number of new nodes inserted into each edge is even. Thus a graph is bipartite if and only if it contains no subgraph that is an even subdivision of $K_3$, the complete graph on three nodes. It is well known that when $G$ is bipartite, $\alpha_w(G) = \bar{\alpha}_w(G)$ for all nonnegative, integral $w$; this follows from total unimodularity of the matrix $A$ for bipartite graphs. Thus $\alpha_w(H) = \bar{\alpha}_w(H)$ for all subgraphs $H$ of $G$ and each nonnegative, integral $w$ if and only if $G$ contains no even subdivision of $K_3$. Here, $H = (W, F)$ is a subgraph of $G = (V, E)$ provided $W \subseteq V$ and $F \subseteq \{\{u, v\} \in E: u, v \in W\}$. Note, in particular, that we do not exclude the case in which $H$ contains an isolated node, as we still have $\alpha_w(H) = \bar{\alpha}_w(H) = +\infty$ in this case.

In this paper we give a similar characterization of graphs $G$ for which $\bar{\alpha}_w(H) - \alpha_w(H) < 1$ holds for all subgraphs $H$ of $G$ and each nonnegative, integral $w$. Note that if $G$ is an even subdivision of $K_4$, then the solution $x_v = \frac{1}{2}$ for all nodes $v$ achieves the value $\bar{\alpha}_1(G) = 2$, but differs from $\alpha_1(G) = 1$ by unity. Similarly, if $G$ consists of two node-disjoint even subdivisions of $K_3$ (i.e., two node-disjoint odd cycles), then again $\bar{\alpha}_1(G) - \alpha_1(G) = 1$. In § 3 we show that $\bar{\alpha}_w(H) - \alpha_w(H) < 1$ for all subgraphs $H$ of $G$ and each nonnegative, integral $w$ if and only if $G$ contains neither of the two configurations just mentioned. We thus obtain a graphical characterization of edge-node incidence matrices having certain "integer rounding" properties (see Trotter [1985]). Linear programming duality shows that the equality $\alpha(G) = \delta(G)$ implies the equality $\alpha_1(G) = \bar{\alpha}_1(G)$, and hence the relation $\bar{\alpha}_w(G) - \alpha_w(G) < 1$ may be viewed as a "near-König" property.

## 2. $\alpha$-Critical graphs.

A graph is *$\alpha$-critical* if, for each edge, its deletion increases the stability number by one. The following important property of $\alpha$-critical graphs was given by Hajnal [1965].

THEOREM 1 (Hajnal). *If $G = (V, E)$ is an $\alpha$-critical graph with no isolated nodes, then the degree of each vertex is at most $|V| - 2\alpha(G) + 1$.*

This theorem can be used to characterize $\alpha$-critical graphs with small values of the parameter $p = |V| - 2\alpha(G)$. Let $\Gamma^p$ be the collection of all $\alpha$-critical graphs such that $p = |V| - 2\alpha(G)$ and let $\Gamma_c^p$ be the collection of all connected graphs in $\Gamma^p$. Let $G \in \Gamma_c^p$. If $p = 0$, then the degree of each node of $G$ is at most one, and hence the graph defined by a single edge is the only member of $\Gamma_c^0$. If $p = 1$, then the degree of each node of $G$ is at most two. The only connected graphs of this type are simple paths and cycles. Now, simple paths and cycles of even length are not $\alpha$-critical. Hence $\Gamma_c^1$ consists of odd cycles. Andrásfai [1967] proved the following theorem for $p = 2$, thus characterizing members of $\Gamma_c^2$.

THEOREM 2 (Andrásfai). *If $G = (V, E)$ is a connected $\alpha$-critical graph with $|V| - 2\alpha(G) = 2$, then $G$ is an even subdivision of $K_4$.*

The collection of all graphs in $\Gamma^2$ without single edges or isolated nodes as components is, by Theorem 2, the collection of all even subdivisions of $K_4$ or graphs consisting of two components, each an odd cycle. Lovász [1978] has established that for larger values of $p$, as well, there is a "finite basis" characterization of the members of $\Gamma_c^p$. That is, for each nonnegative integer $p$, there exists a finite collection of graphs (a *basis*) whose even subdivisions are (precisely) the members of $\Gamma_c^p$. (see Lovász and Plummer [1986, Chap. 12]). The basis is known for $p = 3$ (Lovász [1983]), and we discuss this case in greater detail at the end of § 3.

**3. "Near-König" graphs.** Let $A$ be the edge-node incidence matrix of graph $G = (V, E)$ and let $w = (w_e\colon e \in E)$ be a nonnegative, integral vector. We first state a well-known and useful property (see, e.g., Nemhauser and Trotter [1974]) of the polyhedron of feasible solutions to the linear programming problem that determines $\bar{\alpha}_w(G)$.

LEMMA 1. *Let $x$ be an extreme point of $\{x\colon Ax \leqq w, x \geqq 0\}$. Then $2x$ has integer-valued components. Thus, when $\bar{\alpha}_w(G) < +\infty$, there is an optimum solution $x^*$, i.e., $1 \cdot x^* = \bar{\alpha}_w(G)$, such that $2x^*$ is integer-valued.*

By this lemma we know immediately that $\bar{\alpha}_w(G) - \alpha_w(G)$ is an integer divided by two; when $G$ contains an isolated vertex, i.e., when $\bar{\alpha}_w(G) = \alpha_w(G) = +\infty$, we adopt the convention that $\bar{\alpha}_w(G) - \alpha_w(G) = 0$. Let $\Omega^p$ be the collection of all graphs $G$ such that $\bar{\alpha}_w(H) - \alpha_w(H) < p/2$ for all subgraphs $H$ of $G$ and each nonnegative, integral $w$. Recall that $\Gamma^p$ is defined above to consist of all $\alpha$-critical graphs with $|V| - 2\alpha(G) = p$. We have observed that$\bar{\alpha}_w(H) = \alpha_w(H)$ for all subgraphs $H$ of $G$ and for each nonnegative, integral $w$ if and only if $G$ is bipartite. That is, $G \in \Omega^1$ if and only if $G$ contains no subgraph in $\Gamma^1$. We now show that this relation remains valid for *all* $p > 0$.

LEMMA 2. *Suppose $p$ is a positive integer and $G$ is a graph. Then $G \in \Omega^p$ if and only if $G$ contains no subgraph in $\Gamma^p$.*

*Proof.* Suppose $G' = (V', E')$ is a subgraph of $G$ and $G' \in \Gamma^p$. Since $G' \in \Gamma^p$, we have $|V'| - 2\alpha(G') = p$, and we may remove isolated vertices (if any exist) from $G'$ to obtain a subgraph $H = (W, F)$ of $G$ for which $|W| - 2\alpha(H) \geqq |V'| - 2\alpha(G') = p$. Now $\bar{\alpha}_1(H) < +\infty$ and clearly $\bar{\alpha}_1(H) \geqq |W|/2$, as the solution $x_v = \frac{1}{2}$ for all $v \in W$ is feasible for the problem defining $\bar{\alpha}_1(H)$. Hence $2(\bar{\alpha}_1(H) - \alpha_1(H)) \geqq |W| - 2\alpha_1(H) \geqq p$, and it follows that $G \notin \Omega^p$.

To establish the converse, suppose $G \notin \Omega^p$ and select a subgraph $H = (W, F)$ of $G$ and a nonnegative, integral vector $w$ so that $2(\bar{\alpha}_w(H) - \alpha_w(H)) \geqq p$ with the quantity $|W| + \Sigma(w_e\colon e \in F)$ as small as possible. We will show that $H \in \Gamma^p$. Let $x^*$ denote an optimum linear programming solution yielding value $1 \cdot x^* = \bar{\alpha}_w(H)$; Lemma 1 assures us that we may select $x^*$ so that $2x^*$ is integer-valued.

We first argue that $x_v^* = \frac{1}{2}$, for all $v \in W$, and $w_e = 1$, for all $e \in F$. Consider $\bar{x}$ defined by $\bar{x}_v = \lfloor x_v^* \rfloor$, for all $v \in W$, and define $\bar{w} = w - A\bar{x}$, $\hat{x} = x^* - \bar{x}$, where $A$ is the edge-vertex incidence matrix of $H$. It is straightforward to see that $\hat{x}$ solves $\max\{1 \cdot x\colon Ax \leqq \bar{w}, x \geqq 0\}$. Thus $\bar{\alpha}_{\bar{w}}(H) = \bar{\alpha}_w(H) - 1 \cdot \bar{x}$. Similarly, since $\bar{x}$ is integral, $\alpha_{\bar{w}}(H) + 1 \cdot \bar{x} \leqq \alpha_w(H)$, and hence

$$p/2 \leqq \bar{\alpha}_w(H) - \alpha_w(H) \leqq \bar{\alpha}_{\bar{w}}(H) - \alpha_{\bar{w}}(H).$$

Minimality of $|W| + \Sigma(w_e\colon e \in F)$ and the fact that $\bar{w} \leqq w$ imply that $\bar{w} = w$. Thus $A\bar{x} = 0$ and, as $\bar{\alpha}_w(H) > \alpha_w(H)$ implies $H$ has no isolated nodes, we conclude that $\bar{x} = 0$. It follows that $x_v^* = \hat{x}_v < 1$, for all $v \in W$, which then implies that $w_e \leqq 1$, for all $e \in F$. Furthermore, if $x_v^* = 0$ for some $v \in W$, then $\bar{\alpha}_w(H \backslash v) = \bar{\alpha}_w(H)$ and $\alpha_w(H \backslash v) \leqq \alpha_w(H)$. Thus we have

$$p/2 \leqq \bar{\alpha}_w(H) - \alpha_w(H) \leqq \bar{\alpha}_w(H \backslash v) - \alpha_w(H \backslash v),$$

again contradicting minimality of $|W| + \Sigma(w_e\colon e \in F)$. Thus we must have $x_v^* = \frac{1}{2}$, for all $v \in W$. This also forces $w_e = 1$, for all $e \in F$.

*H* must also be $\alpha$-critical, for if not, $\alpha_1(H \backslash e) = \alpha_1(H)$ for some edge $e \in F$. Since $\bar{\alpha}_1(H \backslash e) \geqq \bar{\alpha}_1(H)$, we would have that

$$p \leqq 2(\bar{\alpha}_1(H) - \alpha_1(H)) \leqq 2(\bar{\alpha}_1(H \backslash e) - \alpha_1(H \backslash e)),$$

in contradiction with the minimality assumption on $H$.

Finally, we show that $|W| - 2\alpha(H) = p$. Since $x_v^* = \frac{1}{2}$, for all $v \in W$, we have $\bar{\alpha}_1(H) = |W|/2$. Hence

$$p \leqq 2(\bar{\alpha}_1(H) - \alpha_1(H)) = 2(|W|/2 - \alpha_1(H)) = |W| - 2\alpha_1(H).$$

To see that $p \geqq |W| - 2\alpha_1(H)$, pick $v \in W$ for which $\alpha_1(H\backslash v) = \alpha_1(H)$. Now $x_v^* = \frac{1}{2}$, for all $v \in W$, implies $\bar{\alpha}_1(H\backslash v) \geqq \bar{\alpha}_1(H) - \frac{1}{2}$, from which it follows that

$$p > 2(\bar{\alpha}_1(H\backslash v) - \alpha_1(H\backslash v)) \geqq 2(\bar{\alpha}_1(H) - \tfrac{1}{2} - \alpha_1(H)) = |W| - 2\alpha_1(H) - 1.$$

Thus $p \geqq |W| - 2\alpha_1(H)$.

We have thus shown that $H$ is an $\alpha$-critical graph for which $|W| - 2\alpha_1(H) = p$. That is, $H \in \Gamma^p$ and the proof is complete. □

Applying Lemma 2 in the case $p = 2$ now gives our main result.

THEOREM 3. *The inequality $\bar{\alpha}_w(H) - \alpha_w(H) < 1$ holds for all subgraphs $H$ of $G$ and each nonnegative, integral $w$ if and only if $G$ contains neither of the following two types of subgraphs*: (i) *two node-disjoint odd cycles*; (ii) *an even subdivision of $K_4$*.

*Proof.* Set $p = 2$ in the definition of $\Omega^p$ above. Then $\Omega^p = \Omega^2$ is exactly the class of graphs such that $\bar{\alpha}_w(H) - \alpha_w(H) < 1$ for all subgraphs $H$ of $G$ and each nonnegative, integral $w$. Thus when $G \in \Omega^2$, Lemma 2 implies that $G$ contains no subgraph in $\Gamma^2$ and hence no subgraph as in (i) and (ii). On the other hand, if $G \notin \Omega^2$, then, as in the proof of Lemma 2, $G$ contains a subgraph in $\Gamma^2$ without single edges or isolated nodes. By the development of the previous section, this subgraph is as in (i) or (ii). □

We may use Theorem 3 to obtain a combinatorial max-min statement relating node-packings and edge-covers in graphs containing neither of the configurations forbidden by the theorem. Specifically, let $G$ be a graph containing neither two node-disjoint odd cycles nor an even subdivision of $K_4$ and let $A$ be the edge-node incidence matrix of $G$. Then, for any nonnegative, integral vector $w$,

$$\begin{aligned}
&\max\,\{1 \cdot x : Ax \leqq w, x \geqq 0, x \text{ integral}\} \\
&\qquad = \lfloor \max\,\{1 \cdot x : Ax \leqq w, x \geqq 0\} \rfloor \quad \text{(by Theorem 3)} \\
&\qquad = \lfloor \tfrac{1}{2} \max\,\{2 \cdot x : Ax \leqq w, x \geqq 0\} \rfloor \quad (\text{where } 2 = (2, \cdots, 2)) \\
&\qquad = \lfloor \tfrac{1}{2} \min\,\{w \cdot y : yA \geqq 2, y \geqq 0\} \rfloor \quad \text{(by linear programming duality)} \\
&\qquad = \lfloor \tfrac{1}{2} \min\,\{w \cdot y : yA \geqq 2, y \geqq 0, y \text{ integral}\} \rfloor,
\end{aligned}$$

where the last equality follows from the fact that the polyhedron $\{y : yA \geqq 1, y \geqq 0\}$ has only $(0, \frac{1}{2}, 1)$-valued extreme points. Thus, e.g., when $G$ has no isolated nodes (otherwise both sides of the max-min relation are $+\infty$), the stability number of $G$ is equal to the "floor" of one-half the value of a 2-cover of nodes by edges in $G$.

If $A$ is any matrix with nonnegative, integral entries, we say that the *integer round-down property* holds for $A$ provided

$$\max\,\{1 \cdot x : Ax \leqq w, x \geqq 0, x \text{ integral}\} = \lfloor \max\,\{1 \cdot x : Ax \leqq w, x \geqq 0\} \rfloor,$$

for all vectors $w$ with nonnegative, integral components (see Baum [1977], Chandrasekaran [1981], Marcotte [1982], Orlin [1982], Trotter [1985], Tipnis [1986]). Thus Theorem 3 characterizes those graphs for which each subgraph has an edge-node incidence matrix satisfying the integer round-down property. Similarly, the *integer round-up property* holds for $A$ when $\min\,\{1 \cdot x : Ax \geqq w, x \geqq 0, x \text{ integral}\} = \lceil \min\,\{1 \cdot x : Ax \geqq w, x \geqq 0\} \rceil$ for all nonnegative, integral $w$. For $G = (V, E)$ with edge-node incidence matrix $A$ and

$w = (1, \cdots, 1)$, the integer programming problem here is known as the *node-covering problem* for $G$. We denote by $\tau(G)$ the value of the node-covering problem for $G$ and say that $G$ is *$\tau$-critical* when the deletion of each edge of $G$ leads to a reduction in $\tau(G)$. Since the (node)-complement of any node-packing in $G$ is a node-cover, and vice-versa, we have that $\alpha(G) + \tau(G) = |V|$ and consequently that $G$ is $\alpha$-critical if and only if $G$ is $\tau$-critical. Thus the characterization given by Andrásfai [1967] in Theorem 2, along with an analysis similar to that provided above, leads to a node-covering analogue of Theorem 3 and the corollary that (precisely) for the class of graphs specified in Theorem 3, each subgraph has an edge-node incidence matrix satisfying the integer round-up property.

We also point out that Chandrasekaran [1981] and Orlin [1982] have given polynomial-time algorithms for solving max $\{1 \cdot x\colon Ax \leqq w, x \geqq 0, x \text{ integer}\}$, where $A$ is a matrix with nonnegative, integral entries and $w$ is any nonnegative, integral vector, provided that $A$ satisfies the integer round-down property. Hence when the graph $G$ is of the type specified in Theorem 3, the node-packing problem for $G$ can be solved in polynomial time. The related recognition question remains open, i.e., whether one can determine in polynomial time that $G$ meets (or fails) the stipulation of Theorem 3.

Finally, we return to the result of Lovász [1983] on $\alpha$-critical graphs for $p = 3$ alluded to at the end of § 2. This characterization leads, in fact, to a result similar to that of Vizing [1964] on the edge-chromatic number of a simple, loopless graph. First consider Vizing's Theorem. A *k-edge-coloring* of a loopless graph $G = (V, E)$ is an assignment of $k$ colors to the edges of $G$ such that no two adjacent edges have the same color. The *edge-chromatic number*, $\chi(G)$, of a graph $G$ is the minimum $k$ such that $G$ has a $k$-edge-coloring. Let $\Delta$ be the maximum degree of a node in $G$. Then Vizing's Theorem asserts that $\chi(G) = \Delta$ or $\chi(G) = \Delta + 1$ for any simple, loopless graph $G$. Now let the rows of $M$ be the (edge-)incidence vectors of matchings in $G$. Then,

$$\chi(G) = \min \{1 \cdot y\colon yM \geqq 1, y \geqq 0, y \text{ integral}\},$$

and it follows from linear programming duality theory that

$$\chi(G) \geqq \min \{1 \cdot y\colon yM \geqq 1, y \geqq 0\} = \max \{1 \cdot x\colon Mx \leqq 1, x \geqq 0\}.$$

Furthermore, $\max \{1 \cdot x\colon \; Mx \leqq 1, \; x \geqq 0\} \geqq \Delta$, since the incidence vector of edges adjacent at a node gives an $x$ satisfying $Mx \leqq 1$, $x \geqq 0$. Thus, Vizing's Theorem implies that the difference between the optimum objective function values of $\min \{1 \cdot y\colon \; yM \geqq 1, \; y \geqq 0, \; y \text{ integral}\}$ and its linear programming relaxation is at most one.

Now, Lovász [1983] has proved that $\Gamma_3^c$ has a basis consisting of the four graphs shown in Fig. 1 and further graphs obtained from these by the following operation: replace a node by two new nodes, each of degree at least one, and create a third new node joined to each of these. Since we know that $\bar{\alpha}_w(G) - \alpha_w(G)$ is an integer divided by two, $\Gamma^3$ can be used in Lemma 2 (i.e., the case $p = 3$) to obtain a characterization of graphs $G$, for which $\bar{\alpha}_w(H) - \alpha_w(H) \leqq 1$ for all subgraphs $H$ of $G$, and each nonnegative, integral $w$.

**Acknowledgments.** Our preparation of this manuscript benefitted from discussions with Gerard Chang and from the comments of anonymous referees. We gratefully acknowledge their assistance.

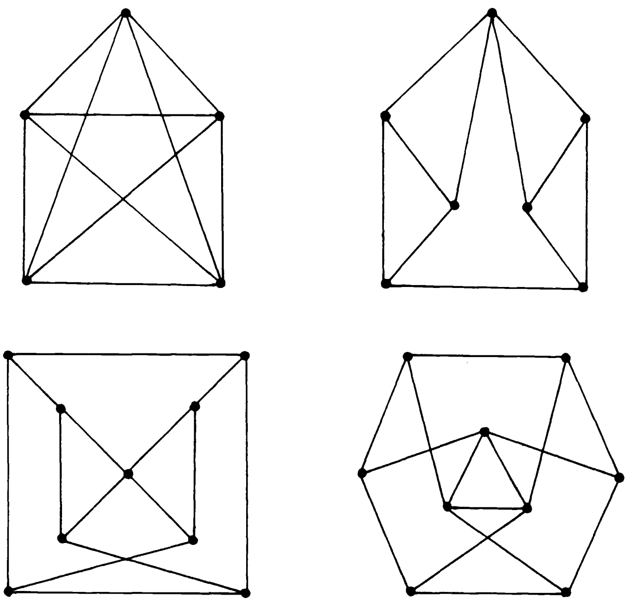


FIG. 1. *Basic graphs in* $\Gamma_c^3$.

## REFERENCES

B. ANDRÁSFAI (1967), *On critical graphs*, in Theory of Graphs, P. Rosenstiehl, ed., Gordon and Breach, New York, pp. 9–19.

S. BAUM (1977), *Integral near-optimal solutions to certain classes of linear programming problems*, Ph.D. Thesis, Technical Report 360, School of OR & IE, Cornell University, Ithaca, New York.

R. CHANDRASEKARAN (1981), *Polynomial algorithms for totally dual integral systems and extensions*, in Studies on Graphs and Discrete Programming, Annals of Discrete Mathematics, 11, P. Hansen, ed., pp. 39–51.

R. W. DEMING (1979), *Independence numbers of graphs—an extension of the König-Egevary Theorem*, Discrete Math., 27, pp. 23–33.

A. HAJNAL (1965), *A theorem on k-saturated graphs*, Canadian Mathematics Journal, 17, pp. 720–724.

E. KORACH (1982), *Packing of T-cuts and other aspects of dual integrality*, Ph.D. Thesis, Department of Combinatorics and Optimization, University of Waterloo, Ontario, Canada.

L. LOVÁSZ (1978), *Some finite basis theorems in graph theory*, in Combinatorics II, A. Hajnal and V. T. Sós, eds., Colloquia Mathematica Societatis János Bolyai, 18, North-Holland, Amsterdam, pp. 717–729.

——— (1983), private communication.

L. LOVÁSZ AND M. D. PLUMMER (1986), *Matching Theory*, Akadémiai Kiadó, Budapest, North-Holland, Amsterdam.

O. M.-C. MARCOTTE (1982), *Topics in combinatorial packing and covering*, Ph.D. Thesis, School of OR & IE, Cornell University, Ithaca, New York.

G. L. NEMHAUSER AND L. E. TROTTER, JR. (1974), *Properties of vertex packing and independence system polyhedra*, Math. Programming, 6, pp. 48–61.

J. B. ORLIN (1982), *A polynomial algorithm for integer programming covering problems satisfying the integer round-up property*, Math. Programming, 22, pp. 231–235.

F. STERBOUL (1979), *A characterization of the graphs in which the transversal number equals the matching number*, J. Combin. Theory Ser. B, 27, pp. 228–229.

S. K. TIPNIS (1986), *Integer rounding and combinatorial* max-min *theorems*, Ph.D. Thesis, Technical Report 680, School of OR & IE, Cornell University, Ithaca, New York.

L. E. TROTTER, JR. (1985), *Discrete packing and covering*, in Combinatorial Optimization, Annotated Bibliographies, M. O'hEigeartaigh, J. K. Lenstra and A. H. G. Rinnooy Kan, eds., Wiley-Interscience, Amsterdam, pp. 21–31.

V. G. VIZING (1964), *On an estimate of the chromatic class of a p-graph*, Diskret. Analysiz., 3, pp. 25–30.

SIAM J. DISC. MATH.
Vol. 2, No. 3, pp. 413–423, August 1989

# GAUSSIAN ELIMINATION WITH PIVOTING IS P-COMPLETE*

STEPHEN A. VAVASIS[†]

**Abstract.** Gaussian elimination with partial pivoting is the standard numerical algorithm for solving unstructured linear systems. Here it is shown that Gaussian elimination with partial pivoting or complete pivoting is log-space complete for P. This provides theoretical evidence that these algorithms cannot be efficiently implemented on a highly parallel computer with a large number of processors. Since other algorithms for linear systems that are efficient on parallel computers are already known, this suggests that elimination-based approaches should not be pursued in a parallel environment with many processors.

**Key words.** Gaussian elimination, pivoting, P-complete, parallel algorithms, computational complexity.

**AMS(MOS) subject classifications.** 65F05, 68Q15

**1. Introduction.** In this paper we show that Gaussian elimination with partial pivoting is P-complete. Gaussian elimination is the oldest and best known method for solving systems of linear equations, and partial pivoting is the most common technique to make elimination numerically stable.

In addition, the proof will be extended to show that complete pivoting is P-complete. Complete pivoting is a less common technique to maintain stability. Partial and complete pivoting are described in the next section. The two pivoting techniques are shown to be P-complete via a log-space reduction from the circuit value problem; the circuit value problem is described in §3. The reduction will show that partial pivoting (but not complete pivoting) is P-complete in the strong sense that the numbers used in the reduction all have bounded size.

These results imply that there is probably no NC algorithm for solving systems of equations based on Gaussian elimination with pivoting. The class of NC languages was first defined by Pippenger [10]. An NC language is accepted by an algorithm running on a parallel computer with a shared memory (the PRAM model) that satisfies the following resource bounds: a polynomial number of processors and a time-bound of $O(\log^k n)$ for a fixed $k$. This time bound, which is asymptotically better than $O(n^\epsilon)$ for any positive $\epsilon$, is the reason why NC algorithms are considered to be efficient in a parallel environment with a large number of processors. P-complete problems are sometimes called "inherently sequential" because if any P-complete problem were in NC, this would imply P=NC, which is generally believed to be false. Thus, the present state of knowledge is that P-complete problems cannot be solved with a parallel polylogarithmic time bound.

The proof below will show that Gaussian elimination with pivoting is a "P-complete algorithm" in the sense that the problem of generating the pivot choices of partial or complete pivoting is P-complete.

* Received by the editors January 21, 1987; accepted for publication (in revised form) November 29, 1988. This work was supported by a Fannie and John Hertz Foundation graduate fellowship which was supplemented by support from the AT & T Foundation, Office of Naval Research contract N00014-85-C-0731, and National Science Foundation grant DCR-8351757. Revision work on this paper was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, Department of Energy, under contract W-31-109-Eng-38 while author was visiting Argonne National Laboratory.

[†] Computer Science Department, Bldg. 460, Stanford University, Stanford, California 94305. Present address, Department of Computer Science, Cornell University, Ithaca, New York 14853.

Anderson and Mayr [2] introduced the notion of P-complete algorithms to explain instances when the steps of a particular algorithm are P-complete even though a different approach to the problem might achieve parallel polylog time.

As a recent example of a different approach to solving linear systems, Pan and Reif [9] proposed a parallel algorithm based on Newton iterations. Their algorithm runs in polylog time and uses $O(n^3)$ processors (fewer if asymptotically efficient matrix multiplication techniques are used).

The search for NC algorithms for linear systems goes back at least to Csanky [4], whose NC algorithm of 1976 was based on subdeterminants.

Thus, solving linear systems is an example where the common sequential algorithm is P-complete even though the underlying problem can be solved in NC using a completely different algorithm. Other examples of this phenomenon have been found in recent years; for example, Reif [11] showed that finding the depth-first search order of a graph that would be found by the standard sequential depth-first search procedure is P-complete, but Aggarwal and Anderson [1] have shown more recently that a depth-first search order (not necessarily the same one) can be produced in randomized NC. The algorithm of Aggarwal and Anderson differs substantially from the simple sequential approach; their algorithm builds the tree up from paths grown all over the graph.

## 2. Gaussian elimination.

In Gaussian elimination, the variables are eliminated one at a time from left to right. The matrix of coefficients $\mathbf{A}$ is transformed to an upper triangular matrix $\mathbf{U}$ by elementary row operations. The multipliers used for elimination may be assembled into a lower triangular matrix $\mathbf{L}$ such that $\mathbf{A} = \mathbf{LU}$.

In plain Gaussian elimination, the elements along the diagonal are used to eliminate the entries in the columns below them. The numbers on the diagonal used for elimination are called *pivots.* A problem arises if zero or a very small number appears in the pivot position. For example, in the following matrix, the first pivot is zero:

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

This matrix is invertible and generally well behaved.

A solution to this problem with the elimination algorithm is to swap the rows of the matrix before each selection of a new pivot. In particular, when the algorithm starts to eliminate entries below the diagonal in column $i$, it first searches all the entries in the column from $i$ downward to find the entry with the largest magnitude, say at position $(r, i)$. Then row $i$ is swapped with row $r$ in order to bring the larger number into the pivot position. This row-swapping technique is called *partial pivoting.*

Intuitively, partial pivoting is a sequential process because the selection of the pivot in column 2 depends on the pivot in column 1, and so forth. In the next section it will be proved that the algorithm is inherently sequential because the sequence of pivot choices can be used to simulate a Boolean circuit.

If all the pivot positions were somehow known in advance, then the Gaussian elimination could be carried out in NC. This is because the rows could all be exchanged to their final positions at the start of the algorithm in a single parallel step ($O(n^2)$ processors working together to move the elements of the matrix). Then each element of the final LU factors of Gaussian elimination can be realized as a large rational function of the original matrix entries. These rational functions can be evaluated quickly in parallel using the technique of Valiant, Skyum, Berkowitz, and Rackoff [13] for polynomial evaluation in tandem with the technique of Strassen [12] to change the rational function into a polynomial. This method for computing LU factors in parallel is discussed by Borodin, von zur Gathen, and Hopcroft [3]. The numerical stability of this approach has not been analyzed.

Partial pivoting is the preferred method for solving general systems of linear equations. Accordingly, it is surprising that the numerical stability of Gaussian elimination with partial pivoting is not completely understood. *Stability* refers to the question of whether roundoff errors introduced by computer arithmetic get magnified into large errors in the results. Although it is possible to construct examples where partial pivoting behaves in an unstable way, years of empirical evidence have shown that partial pivoting is a stable algorithm.

There are good theoretical bounds that show that another technique, *complete pivoting*, is numerically stable. The complete pivoting algorithm swaps both rows and columns in order to bring the largest element in the remaining uneliminated submatrix into the pivot position. This is more costly to implement because of the additional searching and the need to keep track of the permuted variables; consequently, it is used less often. For a discussion of the stability of partial and complete pivoting, as well as detailed listings of the algorithms, see Golub and Van Loan [6, Chap. 4].

## 3. Gaussian elimination with partial pivoting is P-complete.

The main result of this paper is the following theorem.

THEOREM 3.1. *Gaussian elimination with partial pivoting on square matrices over the real or rational numbers is* P*-complete.*

Before stating the proof, we need to explain how Gaussian elimination can be formulated as a language recognition problem. Following are two possible candidates for languages based on elimination.

(a) Given a matrix $\mathbf{A}$ and indices $i$ and $j$, is it true that when Gaussian elimination with partial pivoting is done on $\mathbf{A}$, the pivot used to eliminate the $j$th column entries will be taken from (initial) row $i$?

(b) Given a matrix $\mathbf{A}$ and an index $j$, is it true that when Gaussian elimination with partial pivoting is done on $\mathbf{A}$, the pivot value for column $j$ will be positive?

The following proof will show that both (a) and (b) are P-complete.

*Proof of Theorem* 3.1. The first part of the proof is to show that Gaussian elimination with partial pivoting is in P. The standard algorithm takes $O(n^3)$ arithmetic operations, so the only question about membership in P is the size of the numbers involved. Gaussian elimination with pivoting is in P for exact rational arithmetic as observed by Edmonds [5] because the numerators and denominators of the numbers are determinants of minors. It is also in P for decimal arithmetic rounded to a fixed number of decimal places (a more realistic model) because of bounds on the element growth (see Golub and Van Loan[6]). The reduction that follows will work for either exact arithmetic or arithmetic rounded to two decimal places.

For the proof of P-hardness, we will use a reduction from the circuit value problem. The circuit value problem asks, given an acyclic circuit made of Boolean gates and wires with several inputs and one output, and given the Boolean input values to the circuit, determine the output value. This problem is almost trivial to solve on a sequential computer, but no one knows how to speed it up to polylog time on a parallel computer. This is because the circuit value problem is known to be log-space complete for P, a result due to Ladner [8]. Since log-space reductions are transitive (see Lemma 13.3 of Hopcroft and Ullman [7]), this reduction will show that Gaussian elimination with partial pivoting is also log-space complete for P.

We will assume that the given circuit $C$ is composed entirely of 2-input NAND gates (all other logic gates can be expressed as a fixed combination of NAND gates). Also, we will assume that the NAND gates and the circuit inputs together are numbered $1, \cdots, n$ in a way that is consistent with the partial order of the circuit, and that each gate knows its two fanins and its fanouts. Note that a general circuit can be transformed into a circuit that satisfies all these properties using a simple log-space translator.

In this reduction, each gate of the circuit $C$ will be transformed into two rows and one column of a $2n \times 2n$ matrix $\mathbf{M}_C$. In particular, gate $j$ will be encoded in rows $2j-1$ and $2j$ and in column $j$.

The left half of the matrix (first $n$ columns) is the important half; the other half exists to ensure that the matrix is nonsingular. We want the matrix to be nonsingular to avoid any sort of numerical difficulties, as will be explained below.

A gadget for gate $j$ will appear at matrix elements $(2j-1, j)$ and $(2j, j)$. A gadget for a wire between gate $j$ and $k$ will appear at matrix elements $(2j-1, k)$ and $(2j, k)$, and a second gadget for the wire will appear at entries $(2k-1, j)$ and$(2k, j)$. All entries in the left half of the matrix other than these gadgets will be zeros.

Specifically, the idea of the reduction is that row $2j-1$ of $\mathbf{M}_C$ will be the pivot choice for column $j$ if the output value of gate $j$ in $C$ is 1, or else row $2j$ will be the pivot choice for that column. Gate $j$ will influence other gates via the gadgets for the wires when elimination takes place in column $j$. The two rows $2j-1$ and $2j$ will be called "partners."

For a gate $j$ that is a 1-value circuit input, the contents of positions $(2j-1, j)$ and $(2j, j)$ will be

$$\begin{pmatrix} -3.9 \\ 0 \end{pmatrix}.$$

Note that in this submatrix, the pivot in column $j$ will be taken from row $2j-1$.

For the 0-valued circuit inputs, the contents of positions $(2j-1, j)$ and $(2j, j)$ will be:

$$\begin{pmatrix} -3.9 \\ 4.0 \end{pmatrix}.$$

Observe that this time, the pivot value for column $j$ is in row $2j$.

If $j$ is a NAND gate, the two entries will be:

$$\begin{pmatrix} -3.9 \\ 4.0 \end{pmatrix}.$$

As we shall see in the proof, 4.0 will be chosen as the pivot unless a small constant is subtracted from the second entry as a result of earlier pivots. We will call the entries in column $j$ of rows $2j-1$ and $2j$ the "heavyweight" entries. The gadgets for the wires are as follows: If gate $j$ is an input to gate $k$, then the entries at positions $(2j-1, k)$ and $(2j, k)$ of $\mathbf{M}_C$ looks like this:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

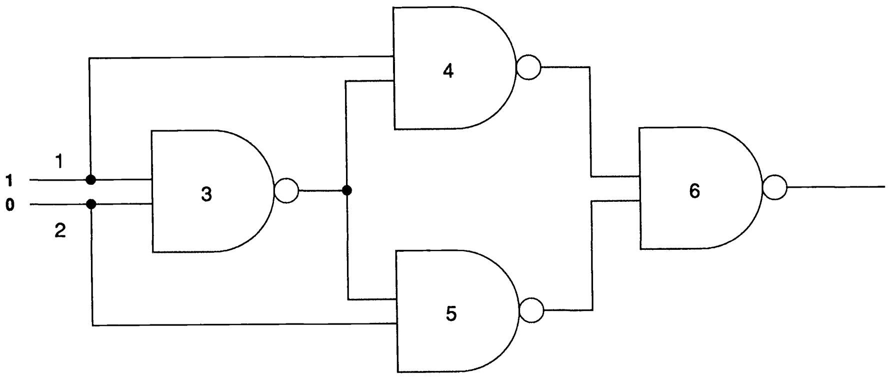


FIG. 1. *This circuit is translated into matrix* $\mathbf{M}_1$.

The plan is that, if row $2j-1$ is selected as the pivot for column $j$, then nothing will be subtracted from the heavyweight entry in position $(2k,k)$ during elimination in column $j$, otherwise the 1 entry, scaled down, will be used during elimination.

If gate $j$ is an input to gate $k$, then matrix entries at positions $(2k-1,j)$ and $(2k,j)$ will look like this:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

These entries force elimination in column $j$ to affect row $2k$.

All entries in the left half of the matrix not covered by the above cases are zeros. Thus, all entries in the left half of the matrix that are not heavyweight entries are 0 or 1 initially. Finally, in the right half of the matrix, the entry at $(2j,n+j)$ will be 1 for all $j$. All other entries of the right half will be zero.

As an example of the reduction, the circuit in Fig. 1 (which happens to be an implementation of XOR) would be transformed to the matrix $\mathbf{M}_1$ below. The rows and columns arenumbered according to the numbers of the gates, and dots indicate zeros. Note that gates 1 and 2 are the circuit inputs. Everything in this transformation can be computed in log-space.

$$\mathbf{M}_1 = \begin{array}{c|cccccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & & & & & & \\
1 & -3.9 & \cdot & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & 0 & \cdot & 1 & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
2 & \cdot & -3.9 & 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & \cdot & 4.0 & 1 & \cdot & 1 & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
3 & 0 & 0 & -3.9 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & 1 & 1 & 4.0 & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
4 & 0 & \cdot & 0 & -3.9 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & 1 & \cdot & 1 & 4.0 & \cdot & 1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\
5 & \cdot & 0 & 0 & \cdot & -3.9 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & \cdot & 1 & 1 & \cdot & 4.0 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\
6 & \cdot & \cdot & \cdot & 0 & 0 & -3.9 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & \cdot & \cdot & \cdot & 1 & 1 & 4.0 & \cdot & \cdot & \cdot & \cdot & \cdot & 1
\end{array}.$$

Informally, when a gate $j$ has an output value of 0 and fans out to gate $k$, the gadget for the wires will cause exactly 0.25 to be subtracted from the '4.0' heavyweight entry in column $k$. This will be enough to force elimination in column $k$ to select the pivot from row $2k-1$ (corresponding to an output value of 1 from gate $k$). Otherwise, the pivot in column $k$ will be selected from row $2k$. This is what we want, since gate $k$ should put out a 1 if and only if one or more of its inputs is a zero.

The fact that this matrix will simulate the circuit will be proved by the following induction lemma. Part (a) is the main step toward proving Theorem 3.1; the other parts are needed to carry the induction along. When we discussed the partial pivoting algorithm earlier, we described it in terms of swapping rows so that the pivots were always in diagonal positions. In the notation of this lemma, however, we will refer to the rows of the matrix $\mathbf{M}_C$ by their original indices for the sake of convenience; accordingly, the pivots will occur in off-diagonal positions. Let $(a_{lm}^{(j)})$ denote the entries of the matrix $\mathbf{M}_C$ after elimination with partial pivoting has been done for columns 1 to $j$ with the rows preserved in their original order.

LEMMA 3.2. *For any $j$ between* 0 *and $n$, after doing the Gaussian elimination on $\mathbf{M}_C$ for columns* 1 *up to $j$, the following statements hold:*

(a) *The pivot in column $j$ will have value $-3.9$ and will be taken from row $2j-1$ if the output of gate $j$ is* 1, *else it will have value* 4.0 *and be taken from row $2j$.*

(b) *All of the columns* 1 *to $j$ have had a pivot in them, and, for each $h \leq j$, one of two partner rows $2h-1$ and $2h$ has had a pivot but not the other.*

(c) *Suppose $l > j$ and gate $l$ is a real* NAND *gate (not a circuit input). Then the heavyweight entry in row $2l$ will satisfy:*

$$a_{2l,l}^{(j)} = 4.0 - 0.25 \cdot t_l^{(j)}.$$

*Here, $t_l^{(j)}$ is the number of* 0*-valued inputs to gate $l$ that are indexed $\leq j$; thus $t_l^{(j)} \in \{0,1,2\}$ always.*

(d) *As above, if $l > j$ and gate $l$ is a real* NAND *gate, then all the entries of row $2l$ in columns $j < q \leq 2n$ will satisfy*

$$|a_{2l,q}^{(j)} - a_{2l,q}^{(0)}| \leq 0.25 \cdot t_l^{(j)}.$$

(e) *Suppose $q \leq 2j$ and row $q$ has not had a pivot; let $l \geq j$. Then if the output of gate $l$ is* 0, *then $|a_{ql}^{(j)}| \leq 1.5$. If the output of gate $l$ is* 1, *then $|a_{ql}^{(j)}| \leq 2.5$.*

*Proof of Lemma* 3.2. First, a general comment about the elimination procedure: observe that, by construction, there is only one nonzero entry in an odd-numbered row $2j-1$, namely, the heavyweight entry in column $j$. Thus, none of the pivots in columns 1 up to $j-1$ can affect this row. Moreover, if we pivot in column $j$ in this row, then it does not affect any matrix entries in columns $j+1$ up to $2n$. These facts will be used several times in the following proof.

The lemma will be proved by induction on $j$. Note that all the statements are clearly satisfied in the case that $j=0$ (for statements (c) and (d), $t_l^{(0)}=0$ for all $l$). Suppose the lemma is true up to $j-1$, and now we must prove the case for $j$. Accordingly, we have pivoted on every column up to $j-1$ and now we are looking to do the pivot in column $j$. Suppose $j$ is a real NAND gate (rather than a circuit input). Consider the entries in column $j$. In rows $2j-1$ and $2j$ are the heavyweight entries; in rows $q < 2j-1$ that have not had a pivot, all the entries are bounded by 2.5 according to (e). All the entries in rows below $2j$ in this column are bounded by 1.5, because (d) tells us that they could not have strayed from their original values (0 or 1) by more than 0.5. Thus, the pivot will be one of the two heavyweight entries in this column. Now, observe that in part (c), all inputs to gate $j$ are indexed $\leq j-1$, so the quantity $t=t_j^{(j-1)}$ will be equal to the number of 0-valued inputs to gate $j$.

If $t=0$ (both inputs to gate $j$ are 1), then (c) tells us that $a_{2j,j}^{(j-1)}=4.0$. Also, $a_{2j-1,j}^{(j-1)}=-3.9$ in all cases because row $2j-1$ is not affected by previous pivots. Thus, pivot will be in row $2j$ with value 4.0. This is the pivot for column $j$; it represents a 0 in the circuit. Note that part (a) is now satisfied since the inputs to this gate were both 1's (we hypothesized $t=0$).

If $t \geq 1$, then by part (c), $0 \leq a_{2j,j}^{(j-1)} \leq 3.75$ but $a_{2j-1,j}^{(j-1)}=-3.9$ so the pivot will be in row $2j-1$ and have a value $-3.9$. Again, part (a) is satisfied. Finally, if gate $j$ is a circuit input, then $a_{2j-1,j}^{(j-1)}=a_{2j-1,j}^{(0)}$ and $a_{2j,j}^{(j-1)}=a_{2j,j}^{(0)}$ because all entries to the left of these were originally zeros, so no pivots in columns up to $j-1$ have affected these rows. Thus, we conclude that part (a) holds by construction for the original entries for gate $j$ when $j$ is a circuit input.

This concludes the induction proof for (a) and (b); now we show that executing the elimination for column $j$ preserves (c) and (d).

First, observe that if we pivoted in row $2j-1$ for column $j$, then no entries in columns $> j$ are affected as mentioned earlier. In addition, $t_l^{(j)}=t_l^{(j-1)}$ for all $l$ in this case by definition of the $t$'s (because pivoting in row $2j-1$ means that the output value of gate $j$ was 1), hence parts (c) and (d) carry through immediately by induction.

Accordingly, we consider the case that we are going to pivot in row $2j$ for column $j$. This means that either $j$ was a "0" circuit input or that gate $j$ was a NAND gate with output value 0, hence with both inputs set to 1. In either case, all the entries in row $2j$ in columns $j$ to $2n$ are equal to their original values by part (d), because $t_j^{(j-1)}=0$.

Examine some row $2l > 2j$. There are two cases to consider: either gate $j$ is an input to gate $l$ with value 0 or gate $j$ is not an input to gate $l$. If $j$ is an input to $l$ with value 0 then the original entry at position $(2j,l)$ was a 1 by construction, so the new entry after step $j-1$ has its original value as mentioned above.

It also happens to be true that the entry at position $(2l,j)$ will be its original value, which is 1. The proof that the $(2l, j)$ entry is unchanged until the $j$th pivot is by contradiction. Suppose that the pivot in column $q$, row $2q$ changed the $(2l, j)$ entry (so $q < j$). Then row $2q$ must have been nonzero in column $j$ when the $q$ pivot was made. By the induction hypothesis (part (d)), if row $2q$ had a pivot then all its entries from $q$ to the right were their original values when the pivot was made. Thus, the only way $2q$ could have a nonzero in column $j$ is if gate $q$ was a "0" input to gate $j$. But this is impossible for a NAND gate, because we are assuming that the output value of gate $j$ is a 0.

Thus, we see that $a_{2j,j}^{(j-1)} = 4.0$, $a_{2j,l}^{(j-1)} = 1$, and $a_{2l,j}^{(j-1)} = 1$. Also, by induction hypothesis on (c),

$$a_{2l,l}^{(j-1)} = 4.0 - 0.25 \cdot t_l^{(j-1)}.$$

Then, by the definition of the Gaussian elimination step,

$$a_{2l,l}^{(j)} = 4.0 - 0.25 \cdot t_l^{(j-1)} - \frac{1 \cdot 1}{4.0}.$$

Since $t_l^{(j)} = t_l^{(j-1)} + 1$ by definition of the $t$'s, this proves statement (c) for the case that $j$ is an input to $l$ of value 0. Note that these calculations are the same in two-place arithmetic.

Let us also do (d) for this case. We have seen that when we pivot in row $2j$, the pivot value is 4.0, the entry at $(2j, l)$ is 1, and all the entries to the right of the pivot are either 0 or 1. This means that, as above, an entry in row $2l$ will change by either 0 or $-0.25$. In either case, the induction on part (d) goes through because $t_l$ increases by 1.

The other case is that $j$ is not an input to gate $l$. The original entry at position $(2l, j)$ was 0 by construction. The claim is that it is still zero when the column $j$ pivot is executed. To see this, suppose some earlier pivot in row $2q$, column $q$ affected it. Then the same contradiction derived above would apply: this would mean $q$ was an input to $j$ of value 0, which contradicts the hypothesis that the output of $j$ is 0. Thus, the entry at position $(2l, j)$ is 0, so pivoting in column $j$ does not affect row $2l$. This completes the induction for (c) and (d).

Now we must do (e), for which we use a more informal induction. Call a row $q$ "abandoned" if it has not had a pivot in columns 1 up to $\lceil q/2 \rceil$. Clearly, once a row is abandoned, we want it to stay that way until the circuit is completely simulated, i.e., until there have been pivots for the first $n$ columns. When a row is first abandoned, all of its entries will be bounded by 1.5; this follows from (d) and the fact that the entries were originally 1 or 0. Let us examine an entry at position $(q, l)$ of an abandoned row. We have seen by the other parts of the lemma that this entry can be affected by pivots in columns 1 to $l - 1$ only if there is an input to gate $l$ of value 0, i.e., only if the output value of gate $l$ is a 1. Thus, if the output of gate $l$ is a 0, entry $(q, l)$ will stay bounded by 1.5, which proves the first half of (e).

If the output of gate $l$ is 1, then pivots in columns 1 to $l - 1$ could affect the $(q, l)$ entry; in particular, for each input $h$ to gate $l$ whose value was 0, the pivot in column $h$ could affect this entry. However, in the previous paragraph we derived the fact that the entry at $(q, h)$ never gets bigger than 1.5 (because the output of $h$ is 0), the pivot value itself is 4.0, and the entry at $(2h, l)$ is either 0 or 1, so the pivot in column $h$ could change $(q, l)$ by at most 1.5/4.0, which is bounded by 0.5. Thus, the entry at $(q, l)$ could never grow past 2.5 because there are at most two pivots that could augment it by 0.5. This proves the second part of (e) and concludes the lemma. Again, note that the computations of the last three paragraphs work out for two-place arithmetic. □

Lemma 3.2 completes the proof of Theorem 3.1, because part (a) shows that after the reduction and the Gaussian elimination, we can determine the value of any gate in the circuit by checking whether a particular pivot was used in a particular column, or whether a particular pivot was positive or negative. This means that we can tell the final value of the circuit based on the position or value of the $n$th pivot choice, so we have reduced the circuit value problem to Gaussian elimination with pivoting.

At this point we want to demonstrate that $\mathbf{M}_C$ is nonsingular because Gaussian elimination on rank-deficient matrices can introduce numerical difficulties if inexact arithmetic is used. We would like to show $\mathbf{M}_C$ is full-rank to demonstrate that the preceding proof does not hinge on a numerical technicality.

This is easily seen by the following observation of Allen Van Gelder: rearrange the rows of $\mathbf{M}_C$ so that the odd numbered rows are written as the top $n$ rows, and the even numbered rows are written as the bottom $n$ rows. Then this modified version of $\mathbf{M}_C$has the form:

$$\begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{X} & \mathbf{I} \end{pmatrix}$$

where $\mathbf{L}$ is diagonal with $-3.9$ on the diagonal, and $\mathbf{I}$ is the identity matrix. This makes it clear that $\mathbf{M}_C$ is nonsingular.

**4. Gaussian elimination with complete pivoting is** P-**complete.** This section will be devoted to extending the proof used in §3 to prove that Gaussian elimination with complete pivoting is P-complete. Either of the two language-recognition versions described above is suitable. The extension is as follows. First, we alter the construction so that instead of $-3.9$ as a heavyweight entry we use $-4.0+\epsilon$ ($\epsilon$ to be chosen below). Analyzing the steps of the above proof tells us that if $0 < \epsilon < 0.25$ then everything works in the same way.

Choose a convenient $0 < \epsilon < \min(0.1, 1/n)$ and define the matrix $\mathbf{M}_C$ that uses $-4.0+\epsilon$ instead of $-3.9$. The proof in §3 still goes through to show that elimination with partial pivoting on this matrix simulates the circuit. Now define $2n \times 2n$ matrix $\mathbf{N}_C$ so that rows $2j-1$ and $2j$ of $\mathbf{N}_C$ are copies of the corresponding rows in matrix $\mathbf{M}_C$ multiplied by the scalar factor $3n-j$. Note that $\mathbf{N}_C$ can easily be constructed in log-space. We claim that the pivots chosen for the first $n$ eliminations in $\mathbf{N}_C$ under complete pivoting will be the same as the pivots chosen in $\mathbf{M}_C$ under partial pivoting. Suppose it is true inductively for the first $j-1$ pivots; this means that the rows of $\mathbf{N}_C$ after $j-1$ pivots (assuming the induction hypothesis) will be the rows of $\mathbf{M}_C$ after $j-1$ pivots multiplied by the scaling factor, because elimination is a linear transformation on the rows. Accordingly, we are now looking for the $j$th pivot. We want the two contenders to be the two heavyweight elements in rows $2j-1$ and $2j$; we know that the magnitude of the larger of these two will be at least $(4.0-\epsilon)(3n-j)$, since the pivot magnitudes in $\mathbf{M}_C$ were always at least $4.0-\epsilon$. If $q < 2j-1$ and row $q$ has not seen a pivot, then we know that the elements in row $q$ of $\mathbf{M}_C$ have maximum magnitude 2.5 by Lemma 3.2(e), so the maximum magnitude of the corresponding elements in $\mathbf{N}_C$ is $2.5(3n-l)$ (where $l$ is the number so that either $q=2l$ or $q=2l-1$). Then we compute that

$$2.5(3n-l) \leq 7.5n$$

$$\begin{aligned} &< 7.8n \\ &< 3.9(3n-n) \\ &< 3.9(3n-j) \\ &< (4.0-\epsilon)(3n-j). \end{aligned}$$

Thus, the two heavyweight elements in rows $2j-1$ and $2j$ dominate everything in previous rows. Suppose now that $q > 2j$; then the largest element in row $q$ is its heavyweight element; in $\mathbf{N}_C$ the magnitude of this entry is bounded by $4.0(3n-l)$, where $l \geq j+1$. We compute:

$$\begin{aligned} 4.0(3n-l) &\leq 4(3n-j-1) \\ &\leq 12n-4j-4 \\ &\leq 12n-4j-3n\epsilon \\ &< (4.0-\epsilon)(3n-j). \end{aligned}$$

Thus, the larger of the heavyweight elements in rows $2j-1$ and $2j$ is the only suitable contender for the $j$th pivot. But they are weighted equally, so whichever would have been picked in $\mathbf{M}_C$ will also be picked in $\mathbf{N}_C$. □

The question of fixed-precision arithmetic becomes murkier in this reduction, because the number of digits needed to write the matrix entries and compute the elimination grows logarithmically with the size of the circuit, instead of staying fixed.

## 5. Conclusions.

These results show that Gaussian elimination with pivoting is P-complete, which suggests that elimination is the wrong approach if one wants an extremely fast algorithm for LU-decomposition on a massively parallel machine. Other techniques deserve consideration in this case.

## 6. Open questions.

We suspect that the algorithm is still P-complete if scaling is used. Scaling, i.e., multiplying the rows or columns by scalars to bring their entries into a desired range, is a technique sometimes used in conjunction with pivoting to improve stability. A new reduction would be needed, however.

There are other numerical algorithms that use interchanges; for example, column interchanges are used in QR-factorization if rank-deficiency is a possibility. This may also lead to a P-complete problem.

Finally, the reduction for the partial pivoting case used all bounded numbers for matrix elements and a fixed number of decimal places, but the reduction for the complete pivoting case needed elements that could grow arbitrarily as the circuit gets large. Is there a reduction in the complete pivoting case that also uses bounded numbers?

**7. Acknowledgments.** The author thanks Shaibal Roy, Alejandro Schäffer, and Ashok Subramanian of Stanford for listening to half-finished versions of this proof, and thanks Professor Ernst Mayr of Stanford for providing much of the background material. Finally, thanks very much to the referee who pointed out a change in my original construction that allowed the reduction of §3 to be carried out with bounded numbers. This greatly simplified the proof and strengthened the result.

## REFERENCES

[1] A. AGGARWAL AND R. J. ANDERSON, *A random* NC *algorithm for depth first search*, Combinatorica, 8 (1988), pp. 1-12.

[2] R. J. ANDERSON AND E. W. MAYR, *Parallelism and greedy algorithms*, Advances in Computing Research, 4 (Parallel and Distributed Computing, 1987), F. P. Preparata, ed., pp. 17-38.

[3] A. BORODIN, J. VON ZUR GATHEN, AND J. HOPCROFT, *Fast parallel matrix and* GCD *computations*, 23rd Symp. Foundations of Computer Science, Chicago, 1982, pp. 65-71.

[4] L. CSANKY, *Fast parallel matrix inversion algorithms*, SIAM J. Comput., 5 (1976), pp. 618-623.

[5] J. EDMONDS, *Systems of Distinct Representatives and Linear Algebra*, J. Res. Nat. Bur. Standards, 71B (1967), pp. 241-245.

[6] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations,* John Hopkins University Press, Baltimore, 1983.

[7] J. E. HOPCROFT AND J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation,* Addison Wesley, Reading, 1979.

[8] R. E. LADNER, *The circuit value problem is log-space complete for* P, SIGACT News, 7:1 (1975), pp. 18-20.

[9] V. PAN AND J. REIF, *Efficient parallel solution of linear systems,* 17th Symp. Theory of Computing, Providence, RI, 1985, pp. 143-152.

[10] N. PIPPENGER, *On simultaneous resource bounds,* 20th Symp. Foundations of Computer Science, San Juan, Puerto Rico, 1979, pp.307-311.

[11] J. H. REIF, *Depth-first search is inherently sequential,* Inform. Process. Lett., 20 (1985), pp. 229-234.

[12] V. STRASSEN, *Vermeidung von Divisionen,* J. Reine. Angew. Math., 264 (1973), pp. 184-202.

[13] L. G. VALIANT, S. SKYUM, S. BERKOWITZ, AND C. RACKOFF, *Fast parallel computation of polynomials using few processors,* SIAM J. Comput., 12 (1983), pp. 641-644.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 425-451, November 1989

# THE ASYMMETRIC ASSIGNMENT PROBLEM AND SOME NEW FACETS OF THE TRAVELING SALESMAN POLYTOPE ON A DIRECTED GRAPH*

EGON BALAS†

**Abstract.** An assignment (spanning union of node-disjoint dicycles) in a directed graph is called asymmetric if it contains at most one arc of each pair $(i, j)$, $(j, i)$. The asymmetric assignment polytope is the convex hull of the incidence vectors of all asymmetric assignments. A class of facets is described for this polytope defined on a complete digraph, associated with certain odd-length closed alternating trails. The inequalities defining these facets are also facet inducing for the traveling salesman polytope defined on the same digraph. Furthermore, this class of facets is distinct from each of the classes identified earlier.

**Key words.** asymmetric assignment problem, traveling salesman polytope (facets of), directed graphs, alternating trails

**AMS(MOS) subject classification.** 05C35

## 1. Introduction.

Let $G = (N, A)$ be the complete digraph on $n = |N|$ nodes with no loops or multiple arcs, and with costs $c_{ij}$ for every arc $(i, j)$. An *assignment* in $G$ is a spanning subgraph that is the node-disjoint union of directed cycles, and the *assignment problem* (AP) is

$$\min \sum (c_{ij}x_{ij} : i,j \in N, i \neq j) \tag{1}$$

$$\text{s.t.} \quad \sum (x_{ij} : j \in N - \{i\}) = 1 \qquad i \in N$$
$$\sum (x_{ij} : i \in N - \{j\}) = 1 \qquad j \in N \tag{2}$$

$$x_{ij} \in \{0, 1\} \qquad i \in N, j \in N - \{i\}. \tag{3}$$

The assignment problem is a frequently used relaxation of the *traveling salesman problem* (TSP), which (on a digraph) asks for a minimum cost directed Hamilton cycle. The TSP on a digraph, often called the *asymmetric traveling salesman problem* (ATSP) to distinguish it from the TSP on an undirected graph, can be stated as having the objective function (1) and a constraint set consisting of (2), (3) and

$$\sum (x_{ij} : i,j \in S, i \neq j) \leqq |S| - 1, \qquad S \subsetneqq N, 2 \leqq |S| \leqq \lfloor n/2 \rfloor. \tag{4}$$

An *asymmetric* assignment is one that contains at most one member of every pair of arcs $(i, j)$, $(j, i)$, i.e., contains no directed 2-cycles. The *asymmetric assignment problem* (AAP) has the objective function (1), and the constraint set (2), (3) and

$$x_{ij} + x_{ji} \leqq 1 \qquad i,j \in N, i \neq j. \tag{5}$$

Clearly, (5) is the subset of (4) corresponding to sets $S \subset N$ such that $|S| = 2$. Thus AAP is also a relaxation of ATSP, stronger (tighter) than AP. In fact, although AAP is closely related to AP, unlike the latter it is NP-complete (see Vornberger [1980] for a proof; see also Garey and Johnson [1980] and Sahni [1974] for a proof of the NP-completeness of the related problem of minimizing (1) subject to (2), (3), and $x_{ij} + x_{kl} \leqq 1$, $\{(i, j), (k, l)\} \in S$, where $S$ is an arbitrary subset of $A \times A$). The *asymmetric*

* Received by the editors June 22, 1987; accepted for publication (in revised form) March 16, 1989.

† Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213. This research was supported by grant ECS 8601660 of the National Science Foundation and contract N00014-85-K-0198 with the United States Office of Naval Research.

*assignment* (AA) *polytope* $P$ is the convex hull of incidence vectors of asymmetric assignments, i.e.,

$$P := \text{conv}\,\{x \in \{0,1\}^{|A|} \mid x \text{ satisfies } (2),(5)\}.$$

An arc set $S \subset A$ that is the node disjoint union of directed cycles and/or paths will be called an (asymmetric) *partial assignment*. If (2′) denotes the system of inequalities obtained from (2) by replacing "=" with "≦", then the incidence vectors of asymmetric partial assignments (APAs for short) are those 0-1 vectors satisfying (2′), (5). The APA *polytope on* $G$ is

$$\tilde{P} := \text{conv}\,\{x \in \{0,1\}^{|A|} \mid x \text{ satisfies } (2'),(5)\},$$

also called the *monotonization of* $P$.

The *asymmetric traveling salesman* (ATS) polytope $P^*$ on $G$ is the convex hull of incidence vectors of tours (directed Hamilton cycles), i.e.,

$$P^* := \text{conv}\,\{x \in \{0,1\}^{|A|} \mid x \text{ satisfies } (2),(4)\}.$$

Finally, the *monotone asymmetric traveling salesman* (MATS) *polytope* $\tilde{P}^*$ is the convex hull of incidence vectors of *partial tours* (arc sets that are subsets of a tour), i.e.,

$$\tilde{P}^* := \text{conv}\,\{x \in \{0,1\}^{|A|} \mid x \text{ satisfies } (2'),(4)\}.$$

The polytope $\tilde{P}$, like $\tilde{P}^*$, is easily seen to be full-dimensional, i.e., $\dim \tilde{P} = \dim \tilde{P}^* = n(n-1)$. As to $P$, since it is contained in the assignment polytope and contains in turn the traveling salesman polytope, and the dimension of these two is known to be the same (Grötschel and Padberg [1985]), namely $n(n-1) - 2n + 1$, it follows that $\dim P = \dim P^* = n(n-1) - 2n + 1$.

In this paper we describe some new classes of facet inducing inequalities for the traveling salesman polytope $P^*$ defined on a directed graph $G$. These inequalities define facets of the asymmetric assignment polytope $P$. They are associated with certain subgraphs of $G$ called closed alternating trails that correspond to odd holes of the intersection graph of the coefficient matrix of the AAP. Section 2 introduces closed alternating trails and establishes their structural properties. Section 3 uses these properties to identify some classes of facet inducing inequalities for $\tilde{P}$ and $\tilde{P}^*$. In § 4 we prove that almost all of these inequalities are also facet inducing for $P$ and $P^*$. Finally, § 5 discusses connections with earlier work.

**2. Closed alternating trails and their chords.** Let $G^* = (V, E)$ be the intersection graph of the coefficient matrix of the system (2), (5) (or the system (2′), (5)). Then $G^*$ has a vertex for every arc of $G$; and two vertices of $G^*$ corresponding, say, to arcs $(p, q)$ and $(r, s)$ of $G$, are joined by an edge of $G^*$ if and only if either $p = r$, or $q = s$, or $p = s$ and $q = r$. Two arcs of $G$ will be called $G^*$*-adjacent* if the corresponding vertices of $G^*$ are adjacent. Clearly, there is a 1-1 correspondence between APA's in $G$ and vertex packings (independent vertex sets) in $G^*$; and therefore the APA polytope $\tilde{P}$ defined on $G$ is identical to the vertex packing polytope defined on $G^*$.

We define an *alternating trail* in $G$ as a sequence of distinct arcs

$$T = (a_1, \cdots, a_t)$$

such that for $k = 1, \cdots, t-1$, $a_k$ and $a_{k+1}$ are $G^*$-adjacent, but $a_k$, $a_l$, $l > k+1$, are not; with the possible exception of $a_1$ and $a_t$. If $a_1$ and $a_t$ are $G^*$-adjacent, then the alternating trail $T$ is *closed*. An arc $a_k = (p, q)$ of $T$ is called *forward* if $p$ is the tail of $a_{k-1}$, or $q$ is the head of $a_{k+1}$, or both; it is called *backward* if $q$ is the head of $a_{k-1}$, or

$p$ is the tail of $a_{k+1}$, or both. The definition of an alternating trail $T$ implies that the direction of the arcs of $T$ alternates between forward and backward, except for pairs $a_k$, $a_{k+1}$ that form a directed 2-cycle entered and exited by $T$ through the same node, in which case $a_k$ and $a_{k+1}$ are both forward or both backward arcs. It also implies that all the 2-cycles of $T$ are node-disjoint (since two 2-cycles of $G$ that share a node define a 4-cycle in $G^*$). Notice that $T$ traverses a node at most twice, and the number of arcs of $T$ incident from (incident to) any node is at most 2. Two alternating trails,

$$T_1 = ((1,2),(3,2),(3,4),(4,3),(5,3),(5,6),(6,5),(6,7))$$

and

$$T_2 = ((2,1),(2,4),(3,4),(3,2),(5,2)),$$

are shown in Fig. 1.

Let $G[T]$ denote the subdigraph of $G$ generated by $T$, i.e., $G[T]$ has $T$ as its arc set and the endpoints of the arcs of $T$ as its node set. Furthermore, for any $v \in N$, let $\deg_T^+(v)$ and $\deg_T^-(v)$ denote the outdegree and indegree, respectively, in $G[T]$ of the node $v$.

The *length* of an alternating trail is the number of its arcs. An alternating trail will be called *even* if it is of even length, *odd* if it is of odd length.

We will be interested in *closed alternating trails* (CATs for short) of odd length. The reason for this is the following proposition.

PROPOSITION 2.1. *There is a* 1-1 *correspondence between odd* CATs *in G and odd holes* (*chordless cycles*) *in $G^*$*.

*Proof.* The proof follows from the definitions. □

It is well known (see Padberg [1973]) that the odd holes of an undirected graph give rise to facets of the vertex packing polytope defined on the subgraph generated by the odd hole, and that these facets in turn can be lifted into facets of the polytope defined on the entire graph (see § 3 for details). In order to make the lifting procedure conveniently applicable to the particular vertex packing polytope associated with $G^*$, we need the structural information concerning adjacency relations on $G^*$ that will be developed in this section.

Let $T$ be a CAT in $G$. A node of $G[T]$ will be called a *source* if it is the common tail of two arcs of $T$ and a *sink* if it is the common head of two arcs of $T$. A node of $G[T]$ can thus be a source, or a sink, or both, or none. A node of $G[T]$ that is neither a source nor a sink will be called *neutral*. A neutral node is incident only with the two

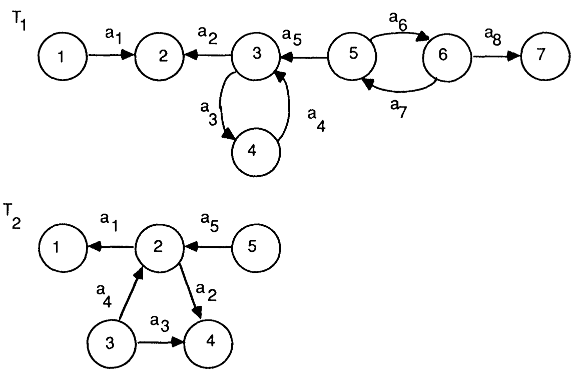


FIG. 1

arcs of a 2-cycle. A 2-cycle will be called *neutral* if it contains a neutral node. Several odd CATs are illustrated in Fig. 2. The sources and sinks of $G[T_1]$ are nodes 1, 2 and 2, 4, respectively, while 3 is neutral. $G[T_2]$ has three neutral nodes, 1, 4, and 6, while nodes 2, 3, and 5 are both sources and sinks. $G[T_3]$ has sources 1 and 4, sinks 2, 3, and 4, while 5 is a neutral node. Further, $G[T_1]$, $G[T_2]$, and $G[T_3]$ have one, three, and one neutral 2-cycles, respectively, and $G[T_3]$ also has a nonneutral 2-cycle.

PROPOSITION 2.2. *Let $T$ be an odd* CAT *of length $t$, with $q$ neutral nodes. Then*

$$1 \leqq q \leqq t/3 \quad \textit{and} \quad q \textit{ is odd}. \tag{6}$$

*Proof.* Let $T = (a_1, \cdots, a_t)$. The alternating sequence of forward and backward arcs is interrupted $q$ times by a repetition of type (forward or backward). Thus for $q$ even, the number of repetitions cancel out, and the fact that $a_1$ and $a_t$ are of opposite directions makes for an even $T$. Since $T$ is odd, $q$ must be odd.

Now suppose $q > t/3$. Then $T$ has more than $2t/3$ arcs incident with neutral nodes, i.e., belonging to 2-cycles entered and exited through the same node, and less than $t/3$ arcs not incident with such nodes. Since $T$ has more than $t/3$ 2-cycles and less than $t/3$ arcs left to separate them, there exists a pair of 2-cycles with a common node. But such a node has indegree and outdegree greater than 2 in $T$, contrary to the definition of an alternating trail. Thus $q \leqq t/3$. □

PROPOSITION 2.3. *Let $T$ be an odd* CAT *of length $t$, with $s$ sources, $u$ sinks, and $w$ 2-cycles. Then*

$$s + u + w = t. \tag{7}$$

*Proof.* $T$ has $2w$ arcs belonging to 2-cycles and $t - 2w$ arcs not belonging to 2-cycles. Every arc of a 2-cycle has either a source for its tail or a sink for its head, but not both. Every arc not belonging to a 2-cycle has both a source for its tail and a sink for its head. Hence

$$\begin{aligned} s+u &= 1 \times w + 2 \times (t-2w)/2 \\ &= t - w. \end{aligned}$$ □

In the following we will denote by $\mathscr{F}$ the family of APAs in $G$.

PROPOSITION 2.4. *Let $T$ be an odd* CAT *of length $t$. Then*

$$\max_{S \in \mathscr{F}} |S \cap T| = (t-1)/2. \tag{8}$$

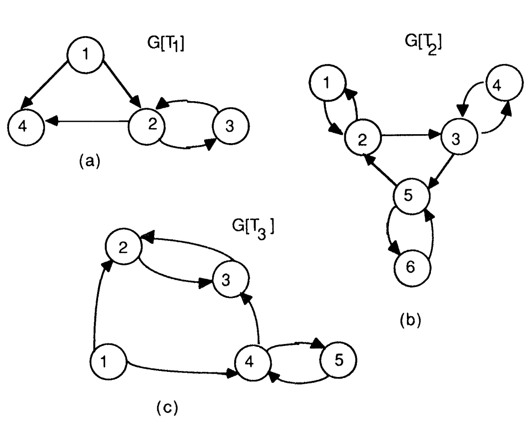


FIG. 2

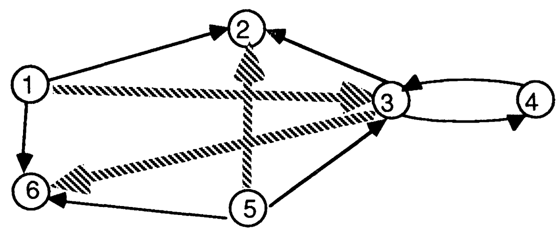


FIG. 3

*Furthermore, for any pair of $G^*$-adjacent arcs $a_k$, $a_{k+1}$ of $T$ (with $t+1=1$), there exists $S \in \mathcal{F}$, with $a_k \notin S$, $a_{k+1} \notin S$, and $|S \cap T| = (t-1)/2$.*

*Proof.* For any $S \in \mathcal{F}$, $S \cap T$ contains no pair of $G^*$-adjacent arcs. The largest such set clearly has cardinality $\lfloor t/2 \rfloor = (t-1)/2$. Furthermore, for any such $S \in \mathcal{F}$, $|T \backslash S| = (t+1)/2$; hence $T \backslash S$ contains a pair of $G^*$-adjacent arcs of $T$; and for any pair $a_k$, $a_{k+1}$ of $G^*$-adjacent arcs of $T$, there exists $S \in \mathcal{F}$ containing $(t-1)/2$ arcs of $T \backslash \{a_k, a_{k+1}\}$. □

A *chord* of a CAT $T$ is an arc $a \in A \backslash T$ joining two nodes of $G[T]$. If $T$ is odd and $a = (u, v)$, $a$ divides $T$ (not always uniquely) into two disjoint subtrails, one odd ($T_1$) and one even ($T_2$), each of which connects $u$ to $v$. We distinguish between three types of chords. A chord $(u, v)$ is of

- *type 1* if it joins a source to a sink (i.e., $\deg_T^+(u) = \deg_T^-(v) = 2$);
- *type 2* if it joins a source to a neutral node, or a neutral node to a sink (i.e., $\deg_T^+(u) + \deg_T^-(v) = 3$), and there exists an *even* subtrail $T_2$ connecting $u$ to $v$ whose first arc and last arc are forward arcs;
- *type 3* in all other cases.

Figure 3 shows the odd CAT

$$T_1 = ((1,2),(3,2),(3,4),(4,3),(5,3),(5,6),(1,6))$$

with its chords of type 1 $(1, 3)$, $(3, 6)$, $(5, 2)$ in shaded lines.

As $T_1$ has no chords of type 2, all other chords (not shown) are of type 3.

Figure 4 shows the odd CAT

$$T_2 = ((1,2),(3,2),(2,3),(2,4),(5,4),(4,5),(4,6),(7,6),(6,7),(6,8),(1,8))$$

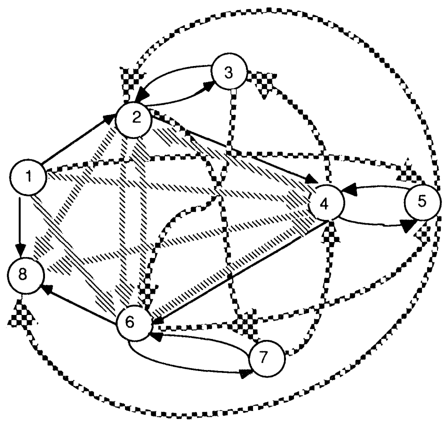


FIG. 4

with its chords of type 1 in shaded lines, ((1, 4), (1, 6), (2, 6), (2, 8), (4, 2), (4, 8), (6, 2), (6, 4)), and its chords of type 2 in checkered lines, ((1, 5), (2, 7), (3, 6), (4, 3), (5, 2), (5, 8), (6, 5) and (7, 4)). All other chords (not shown) are of type 3.

Finally, of the three odd CATs of Fig. 2, $T_1$ has only chords of type 3; $T_2$ has three chords of type 1, (2, 5), (3, 2), and (5, 3), and three of type 2, (1, 5), (4, 2), and (6, 3); and $T_3$ has two chords of type 1, (1, 3) and (4, 2); all remaining chords are of type 3.

The chords of type 1 are easily identified. To illustrate the identification of chords of type 2, the chord (7, 4) of the odd CAT of Fig. 4 is of type 2 since the subtrail $\{(7,6),(4,6),(4,5),(5,4)\}$ connecting 7 to 4 is even, and both (7, 6) and (5, 4) are forward arcs. Similarly, the subtrails whose presence makes (6, 5) and (4, 3) chords of type 2, namely $\{(6,7),(7,6),(4,6),(4,5)\}$ and $\{(4,5),(5,4),(2,4),(2,3)\}$, have the required properties. On the other hand, in the odd CAT of Fig. 2(c), the chord (5, 2) is not of type 2, although it joins the neutral node 5 to the sink 2, and although there exists an even subtrail $\{(5,4),(1,4),(1,2),(3,2)\}$ that connects 5 to 2. This is because the last arc of this even subtrail, (3, 2), is reverse, and there exists no subtrail with the required properties.

As before, $\mathscr{F}$ denotes the family of APAs in $G$.

PROPOSITION 2.5. *Let $T$ be an odd* CAT *of length $t$, let $C$ be the set of chords of $T$, and for $k$* = 1, 2, 3, *let $C_k$ be the set of chords of $T$ of type $k$. Then*

$$(9)\qquad \max_{S\in\mathscr{F}} |S\cap(T\cup\{a\})| = \begin{cases} (t-1)/2 & a\in C_1\cup C_2 \\ (t+1)/2 & a\in C_3\cup(A\backslash(T\cup C)). \end{cases}$$

*Proof.* Let $a=(u,v)$, and let $T=T_1\cup T_2$, where $T_1$ and $T_2$ are two disjoint subtrails of $T$ connecting $u$ to $v$, with $T_1$ odd and $T_2$ even. If $a\notin S$, then $S\cap(T\cup\{a\}) = S\cap T$ and from Proposition 2.4 the left-hand side of (9) is equal to $(t-1)/2$. So assume $a\in S$. If $a\in C_1$, this implies that $S$ cannot contain the first and last arcs of $T_1$ and $T_2$. Hence the maximum number of arcs of $T_1$ and $T_2$ contained by any such $S$ is $(|T_1|-1)/2$ and $(|T_2|-2)/2 = |T_2|/2-1$, respectively, and the maximum of the expression on the left-hand side of (9) is $(|T_1|-1)/2$ (arcs of $T_1$) + $|T_2|/2-1$ (arcs of $T_2$) + 1 (the arc $a$) $=(|T_1|+|T_2|-1)/2=(t-1)/2$.

If $a\in C_2$, then $a\in S$ implies that $S$ cannot contain the first and the last arc of $T_2$ (by definition of $C_2$, the last arc of $T_2$ is incident *to $v$*) and $S$ cannot contain *both* the first and last arcs of $T_1$. Then the maximum number of arcs of $T_2$ contained by any $S\in\mathscr{F}$ that contains $a$ is, as in the earlier case, $|T_2|/2-1$, and the maximum number of arcs of $T_1$ is $(|T_1|-1)/2$, also like in the earlier case. Thus the left-hand side of (9) is again equal to $(t-1)/2$.

If $a\in C_3$, $S$ can contain, besides $a$, $(|T_1|-1)/2$ arcs of $T_1$ and $|T_2|/2$ arcs of $T_2$, and the maximum of the left-hand side of (9) is $(|T_1|-1)/2+|T_2|/2+1 = (t+1)/2$.

Finally, if $a\in A\backslash(T\cup C)$, then $S$ can contain, besides $a$, $(|T|-1)/2$ arcs of $T$, hence $(|T|-1)/2+1=(t+1)/2$ arcs. □

PROPOSITION 2.6. *Let $T$ be an odd* CAT *of length $t$ and let $C_1$ be the set of chords of $T$ of type* 1. *Then*

$$(10)\qquad \max_{S\in\mathscr{F}} |S\cap(T\cup C_1)| = (t-1)/2.$$

*Proof.* Let $S^*\in\mathscr{F}$ be an APA that maximizes $|S\cap(T\cup C_1)|$. Without loss of generality, we may assume that for every neutral node $v$ of $G[T]$, $S^*\cap T$ contains an arc incident with $v$. Indeed, should this not be the case for some $v$, one can always replace the arc of $S^* \cap T$ incident with the (unique) node adjacent in $T$ to $v$, with one of the two arcs incident with $v$ in order to obtain an APA $S^0$ such that $|S^0 \cap (T \cup C_1)| = |S^* \cap (T \cap C_1)|$.

Let $s$, $u$, and $q$ denote the number of sources, sinks, and neutral nodes, respectively, of $G[T]$, and let $w$ stand for the number of 2-cycles of $T$. Let $S^* \cap (T \cup C_1) = S_1 \cup S_2$, where $S_1$ is the set of arcs in $S^* \cap$ T incident with a neutral node, and $S_2 = S^* \cap (T \cup C_1) \backslash S_1$. By assumption, $|S_1| = q$. From Proposition 2.3 and the fact that $q \leqq w$,

$$s + u \leqq t - q.$$

Every arc in $S_2$ has either a source for its tail, or a sink for its head, or both; every arc in $S_1$ has either a source for its tail or a sink for its head, but not both; and no two arcs in $S_1 \cup S_2$ have a common tail or a common head. Hence

$$\begin{aligned} |S_1 \cup S_2| &\leqq q + \tfrac{1}{2}(s+u-q) \\ &\leqq q + \tfrac{1}{2}(t-2q) \\ &= \tfrac{1}{2}t \end{aligned}$$

or, since $t$ is odd, $|S^* \cap (T \cup C_1)| \leqq (t-1)/2$. □

PROPOSITION 2.7. *Let $T$ be an odd* CAT *of length $t$, and let $C_1$ be the set of chords of $T$ of type* 1. *Then*

$$(11) \qquad \max_{S \in \mathcal{F}} |S \cap (T \cup C_1 \cup \{a\})| = (t+1)/2, \quad \forall a \in A \backslash (T \cup C_1).$$

*Proof.* From Proposition 2.5, for $a \in A \backslash (T \cup C_1 \cup C_2)$ the left-hand side of (11) is $\geqq (t+1)/2$. If the inequality is strict for some $S$, that implies that $S$ contains $(t+1)/2$ arcs of $T \cup C_1$, contrary to Proposition 2.6. Thus (11) holds for $a \in A \backslash (T \cup C_1 \cup C_2)$.

Now let $a \in C_2$. Without loss of generality, assume that $v$ is neutral (an analogous reasoning holds if $u$ is neutral). Then the last arc of $T_1$, say $a_p$, is incident from $v$. We put $a_p$ into $S$ and consider two cases.

*Case* 1. The first two arcs of $T_2$ do not form a 2-cycle. Then they have a common head $w$. We put into $S$ all arcs of $T_2$ whose position in the sequence $T_2$ is odd and $\geqq 3$. Then $\deg_S^-(w) = 0$ and $|S \cap T_2| = |T_2|/2 - 1$. Since no pair of the last three arcs $a_{p-2}$, $a_{p-1}$, $a_p$ of $T_1$ form a 2-cycle, and since $a_p$ is a reverse arc, $a_{p-2}$ and $a_{p-1}$ have a common tail $z$. We put in $S$ all arcs of $T_1$ whose rank in $T_1$ is even and $\leqq p-3$. Then $\deg_S^+(z) = 0$ and $|S \cap T_1| = (|T_1| - 1)/2$. Now $(z, w)$ joins a source to a sink; hence it is a chord of type 1. Putting $(z, w)$ in $S$ raises the cardinality of $S$ to $(|T_1| - 1)/2$ (arcs of $T_1$) + $(|T_2|/2) - 1$ (arcs of $T_2$) + 1 (the chord $(z, w) \in C_1$) + 1 (the chord $a \in C_2$) = $(t+1)/2$.

*Case* 2. The first two arcs of $T_1$ do not form a 2-cycle. Then they have a common head $w$. We put into $S$ all arcs of $T_1$ whose rank in $T_1$ is odd and $\geqq 3$. Since the last two arcs of $T_2$ do not form a 2-cycle, they have a common tail $z$. We put into $S$ all arcs of $T_2$ whose rank in $T_2$ is even and $\leqq |T_2| - 2$. Putting into $S$ the chord $(z, w) \in C_1$ again yields a set of cardinality $(t+1)/2$. □

So far we have assumed that $G$ is complete. The following result, which relaxes this assumption, will be needed in the next section. For a CAT $T$ in a directed (not necessarily complete) digraph $G = (N, A)$, we will say that $T$ is $C_1$-*complete* if every source of $T$ is joined to every sink of $T$ by an arc in $A$.

COROLLARY 2.8. *Proposition* 2.7 *holds if $G$ is not complete, but $T$ is $C_1$-complete.*

*Proof.* Obvious from the proof of Proposition 2.7. □

**3. Facets of the monotone polytopes $\tilde{P}$ and $\tilde{P}^*$.** We are now ready to characterize the class of facet inducing inequalities of the APA polytope $\tilde{P}$ associated with odd CATs. We consider first the subgraph generated by an odd CAT.

As mentioned in § 2, $\tilde{P}$ is the same as the vertex packing polytope defined on $G^*$. Furthermore, every odd CAT of $G$ corresponds to an odd hole of $G^*$. It is well known (Padberg [1973]) that odd holes of an undirected graph give rise to facet inducing inequalities for the vertex packing polyhedron defined on the subgraph generated by the odd hole. Nevertheless, because of its simplicity and its usefulness in subsequent developments, we give a direct proof of this result for our case.

For any $S \subseteq A$, we denote $x(S) = \sum (x_{ij} : (i, j) \in S)$.

PROPOSITION 3.1. *Let $T$ be an odd* CAT *of length $t$, and let $\tilde{P}(G[T])$ be the* APA *polytope defined on $G[T]$. Then the inequality*

$$(12) \qquad x(T) \leqq (t-1)/2$$

*defines a facet of $\tilde{P}(G[T])$.*

*Proof.* From Proposition 2.4, (12) is satisfied by all $x \in \tilde{P}(G[T])$. Let $T := (a_1, \cdots, a_t)$. Define $x^1 \in \tilde{P}(G[T])$ by $x^1_{a_i} = 1$ if $i$ is odd and $1 \leqq i < t$, $x^1_{a_i} = 0$ otherwise, and for $k = 2, \cdots, t$, define $x^k$ by $x^k_{a_i} = x^{k-1}_{a_{i-1}}$, $i = 1, \cdots, t$, with $i - 1 = t$ for $i = 1$. Then the vectors $x^k$, $k = 1, \cdots, t$ form the rows of a circulant matrix of order $t$ with $(t-1)/2$ 1's in every row (and every column) known to be nonsingular. Hence the $t$ points $x^k$, $k = 1, \cdots, t$, which are clearly contained in

$$\tilde{P}(G[T]) \cap \{x | x(T) = (t-1)/2\},$$

are affinely independent. Thus (12) defines a facet of $\tilde{P}(G[T])$. □

COROLLARY 3.2. *The inequality* (12) *defines a facet of the* MATS *polytope $\tilde{P}^*(G[T])$.*

*Proof.* Since $P^*(G[T]) \subset P(G[T])$, the inequality (12) is valid for $\tilde{P}^*(G[T])$. Since $\tilde{P}^*(G[T])$ is full-dimensional, (12) does not define an improper face. Finally, each of the $t$ affinely independent points $x^k \in \tilde{P}(G[T])$ used in the proof of Proposition 3.1 is a point of $\tilde{P}^*(G[T])$, i.e., an incidence vector of a partial tour. Thus (12) defines a facet of $\tilde{P}^*(G[T])$. □

Next we will "lift" the inequality (12) to identify inequalities of the form

$$(13) \qquad x(T) + \sum (\alpha_{ij} x_{ij} : (i,j) \in A \setminus T) \leqq (t-1)/2$$

that define facets of $\tilde{P}$. It is a well-known result in combinatorial optimization (see Padberg [1973], Nemhauser and Trotter [1974], Balas and Zemel [1984]) that if (12) defines a facet of $\tilde{P}(G[T])$, then there exist integers $\alpha_{ij}$, $(i, j) \in A \setminus T$, such that (13) defines a facet of $\tilde{P}$. Furthermore, for any ordering $(i(1), j(1)), \cdots, (i(p), j(p))$ of the arc set $A \setminus T$, there exists such a facet defining inequality, whose coefficients $\alpha_{ij}$ can be obtained by solving a sequence of integer programs. To be more specific, if for $k = 1, \cdots, p$, $G_k = (N_k, A_k)$ is the graph consisting of the arcs in

$$T \cup \{((i(1), j(1)), \cdots, (i(k), j(k))\}$$

and their endpoints, the coefficients of (13) are obtained by setting, for $k = 1, \cdots, p$,

$$(14) \qquad \alpha_{i(k)j(k)} = (t-1)/2 - z_{i(k)j(k)},$$

where

$$
\begin{aligned}
z_{i(k)j(k)} = \max\ & \sum (x_{ij} : (i,j) \in A_k \setminus \{(i(k), j(k))\}) \\
& \sum (x_{ij} : j \in \Gamma_k(i) \leqq 1 \qquad i \in N_k \setminus \{i(k)\} \\
(15) \qquad & \sum (x_{ij} : i \in \Gamma_k^{-1}(j)) \leqq 1 \qquad j \in N_k \setminus \{j(k)\} \\
& x_{i(k)j} = 0, \quad j \neq j(k); x_{ij(k)} = 0, \quad i \neq i(k) \\
& x_{ij} \in \{0, 1\}, \qquad (i,j) \in A_k,
\end{aligned}
$$

and where $\Gamma_k(i)$ and $\Gamma_k^{-1}(i)$ are the sets of successors and predecessors, respectively, of node $i$ in $G_k$.

It follows from the above definition of the lifting coefficients that in comparing two inequalities of the form (13), say $(13)_1$ with coefficients $\alpha_{ij}^1$, and $(13)_2$ with coefficients $\alpha_{ij}^2$, for any given arc $(i_*, j_*)$ we have $\alpha_{i_*j_*}^1 \geqq \alpha_{i_*j_*}^2$ if the rank of $(i_*, j_*)$ in the sequence associated with $(13)_1$ is lower than in the sequence associated with $(13)_2$. Therefore, a given coefficient has the highest value if the corresponding variable is lifted first and the lowest value if it is lifted last.

THEOREM 3.3. *Let $G = (N, A)$ be a complete digraph. Let $T$ be an odd* CAT *of length $t$, and let $C_1$ be the set of chords of $T$ of type* 1. *Then the inequality*

$$(16) \qquad x(T \cup C_1) \leqq (t-1)/2$$

*defines a facet of the* APA *polytope $\tilde{P}$ and the* MATS *polytope $\tilde{P}^*$.*

*Proof.* We lift the inequality (12) by taking the arcs of $A \setminus T$ in any order such that all arcs in $C_1$ precede all arcs in $A \setminus (T \cup C_1)$. Let $c_1 = (i(1), j(1))$ be the arc in $C_1$ whose variable is lifted first. Then the maximum of the integer program (15) is $z_{i(1)j(1)} = (t-3)/2$, since from Proposition 2.5

$$\max_{S \in \mathcal{F} : c_1 \in S} |S \cap T| = \max_{S \in \mathcal{F} : c_1 \in S} |S \cap (T \cup \{c_1\})| - 1 = (t-3)/2.$$

Thus $\alpha_{i(1)j(1)} = (t-1)/2 - z_{i(1)j(1)} = 1$.

We claim that $\alpha_{i(k)j(k)} = 1$, $k = 1, \cdots, m$, where

$$\{(i(1), j(1)), \cdots, (i(m), j(m))\} = C_1.$$

Suppose the claim is true for $k = 1, \cdots, l-1$, and let $k = l \geqq 2$. From Propositions 2.5 and 2.6,

$$
\begin{aligned}
&\max \{ |S \cap (T \cup \{(i(1), j(1)), \cdots, (i(l), j(l))| \} \\
&\qquad = \max \{ |S \cap (T \cup \{(i(1), j(1)), \cdots, (i(l), j(l))| : (i(l), j(l)) \in S \} = (t-1)/2,
\end{aligned}
$$

hence

$$
\begin{aligned}
z_{i(l)j(l)} &= \max \{ |S \cap (T \cup \{i(1), j(1), \cdots, (i(l-1), j(l-1))| : (i(l), j(l)) \in S \} \\
&= (t-3)/2,
\end{aligned}
$$

and thus $\alpha_{i(l)j(l)} = (t-1)/2 - z_{i(l)j(l)} = 1$, which proves the claim. Thus the lifting coefficients of all variables corresponding to arcs of $C_1$ are equal to 1.

Consider now the coefficient of the variable associated with some arc $a \in A \setminus (T \cup C_1)$ that is lifted *first* after the variables corresponding to arcs in $C_1$. Let $a =$

$(i(m+1), j(m+1))$. Since $T$ is $C_1$-complete in the graph $G_{m+1}$, from Corollary 2.8 we have

$$\begin{aligned}&\max\{|S\cap(T\cup\{(i(1),j(1)),\cdots,(i(m+1),j(m+1))|\}\\&=\max\{|S\cap(T\cup\{(i(1),j(1)),\cdots,(i(m+1),j(m+1))|:(i(m+1),j(m+1))\in S\}\\&=(t+1)/2,\end{aligned}$$

and thus

$$\begin{aligned}z_{i(m+1)j(m+1)}=\max\{|S\cap(T\cup\{(i(1),j(1)),\cdots,(i(m),j(m))|:&\\(i(m+1),j(m+1))\in S\}&\\=(t+1)/2-1=(t-1)/2.&\end{aligned}$$

Therefore, $\alpha_{i(m+1)j(m+1)}=(t-1)/2-z_{i(m+1)j(m+1)}=0$. Since the variable associated with the arc $a$ has a coefficient of 0 when it is first in the lifting sequence (among the arcs in $A\backslash(T\cup C_1)$), it has a coefficient of 0 also when it is in any subsequent position in the sequence.

This proves that the lifting coefficients of the arcs $a\in A\backslash T$ are, for *any* lifting sequence that puts all arcs in $C_1$ before all arcs in $A\backslash(T\cup C_1)$,

$$\alpha_a=\begin{cases}1 & a\in C_1\\ 0 & a\in A\backslash(T\cup C_1)\end{cases}$$

which proves that (16) defines a facet of $\tilde{P}$ and of $\tilde{P}^*$. □

The arc sets corresponding to the support (i.e., the set of positive coefficients) of each inequality (16) in the digraphs with four, five, and six vertices are shown (up to isomorphism) in Figs. 5, 6, and 7, with the arcs of $T$ and $C_1$ shown in solid and shaded lines, respectively. The number of such inequalities is growing faster than exponentially with $n$.

It is easy to establish the *Chvatal rank* of the inequalities (16). Chvatal's [1973a] procedure for generating all the inequalities valid for a polyhedron defined as the convex hull of integer points satisfying a given set of linear inequalities $Ax \leqq b$ consists of recursively applying the following step: take all undominated positive linear combinations of the inequalities of the current system and add the resulting inequalities to the system after rounding down all coefficients to the nearest integer. The initial system $Ax \leqq b$ is said to have rank 0, while the inequalities obtained in the first step of the recursion have rank 1.

*Remark* 3.4. The inequalities (16) have Chvatal rank 1.

*Proof*. Each inequality (16) associated with an odd CAT $T$ can be obtained by adding the equations (2) associated with each source and each sink of $G[T]$ and the

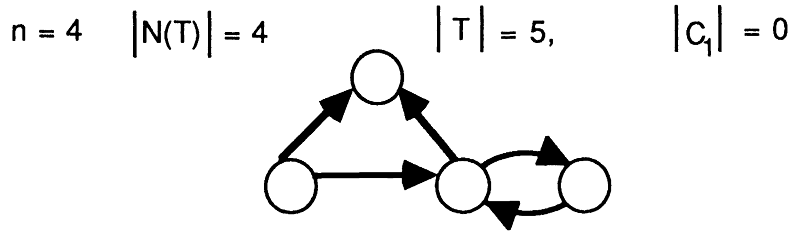


FIG. 5

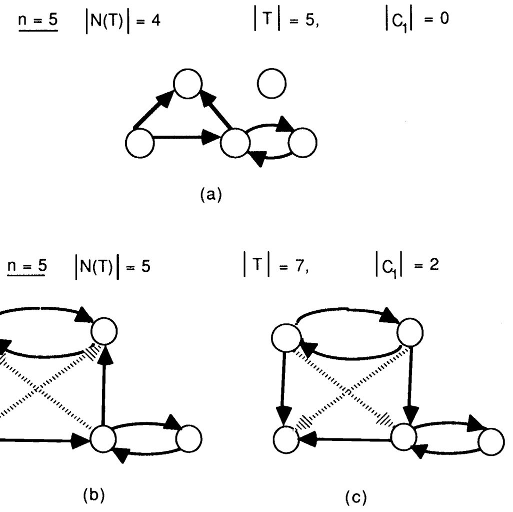

n = 5 |N(T)| = 4 |T| = 5, |C1| = 0
(a)
n = 5 |N(T)| = 5 |T| = 7, |C1| = 2
(b)
(c)

FIG. 6

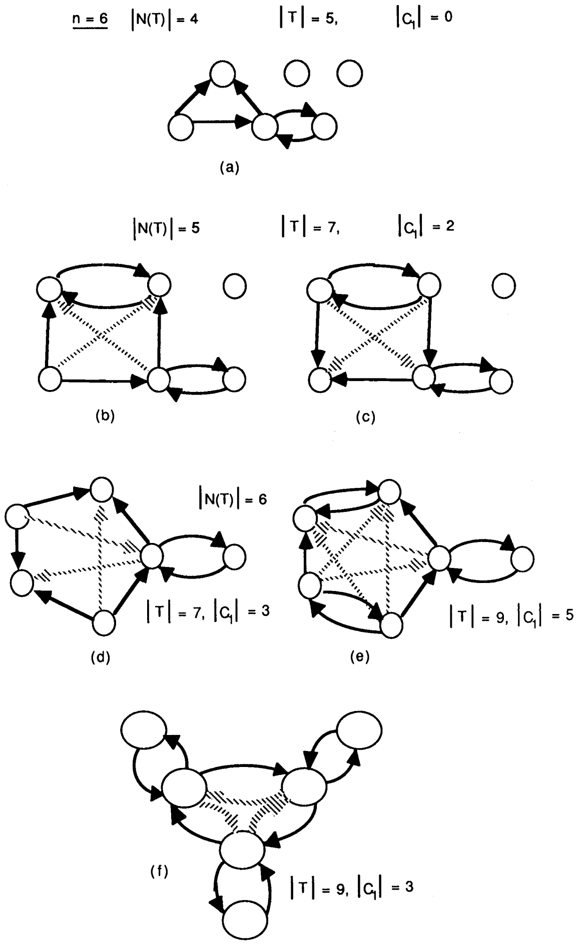

n = 6 |N(T)| = 4 |T| = 5, |C1| = 0
(a)
|N(T)| = 5 |T| = 7, |C1| = 2
(b)
(c)
|N(T)| = 6
|T| = 7, |C1| = 3
(d)
|T| = 9, |C1| = 5
(e)
(f)
|T| = 9, |C1| = 3

FIG. 7

inequalities associated with each two-cycle of $G[T]$; then dividing the resulting inequality by two and rounding down all coefficients to the nearest integer. □

There are some inequalities other than the family (16) that can be obtained by lifting the inequality (12), but their description is more cumbersome. The following rules apply to all facet-defining inequalities (for $\tilde{P}$ and $\tilde{P}^*$) obtained by sequential lifting from (12). As before, for $k = 1, 2, 3$, let $C_k$ be the set of chords of type $k$.

- All variables corresponding to chords in $C_1 \cup C_2$ get a coefficient of 0 or 1, and all remaining variables get a coefficient of 0, irrespective of the lifting sequence.
- If all variables corresponding to chords in $C_1$ are lifted before all others, then all variables corresponding to chords in $C_1$ get a coefficient of 1 and all remaining variables get a coefficient of 0 (this is the family (16)).
- If a variable corresponding to a chord $(i, j) \in C_2$ is lifted first, it gets a coefficient of 1; but then the variables corresponding to certain chords in $C_1$, whose identity depends on $(i, j)$, get a coefficient of 0, as do all the variables corresponding to arcs in $A \backslash (T \cup C_1 \cup \{(i, j)\})$.

**4. Facets of the polytopes P and P*.** In this section we will prove that, with the exception of one special case for $n = 5$ and one for $n = 6$, the inequality (16) defines a facet of the ATS polytope $P^*$. It then follows that it also defines a facet of the AA polytope $P$.

For the sake of brevity, the incidence vector of a tour will sometimes be called a tour. A tour (or its incidence vector) will be called *extreme* with respect to some valid inequality if it satisfies with equality the inequality in question.

First we establish a useful property of all nontrivial facet-defining inequalities for $P^*$ (i.e., inequalities different from $x_{ij} \geqq 0$ and $x_{ij} \leqq 1$). Let $\alpha x \leqq \alpha_0$ be a nontrivial valid inequality for $P^*$, and let $X$ be the matrix whose rows are the incidence vectors $x$ of tours in $G$ satisfying $\alpha x = \alpha_0$. If $\alpha x \leqq \alpha_0$ defines a facet of $P^*$, then for every arc $(i, j)$ of $G$ there exists some extreme tour $x$ such that $x_{ij} = 1$, and some extreme tour $x'$ such that $x'_{ij} = 0$. Therefore the matrix $X$ has the following two obvious properties:

(i) every column of $X$ contains at least one 0 and at least one 1;

(ii) for every $p \in N$, the set of $n - 1$ columns $(i, p)$, $i \in N \backslash \{p\}$, and the set of $n - 1$ columns $(p, j)$, $j \in N \backslash \{p\}$, form two submatrices each of which has rank $n - 1$.

A less obvious but more useful property of $X$ is the following.

THEOREM 4.1. *Suppose $\alpha x \leqq \alpha_0$ defines a facet of $P^*$. For any $p$, $q \in N$, let $A(p, q)$ be the set of columns $(i, p)$, $i \in N \backslash \{p, q\}$, and $(q, j)$, $j \in N \backslash \{p, q\}$, of $X$. Then the submatrix of $X$ consisting of the columns in $A(p, q)$ has rank $|A(p, q)| - 1$;* i.e., $2n - 3$ *if $p = q$ and* $2n - 5$ *if $p \neq q$.*

*Proof.* Let $X(p, q)$ be the submatrix of $X$ consisting of the columns in $A(p, q)$. Clearly, $X(p, q)$ has $2n - 2$ columns if $p = q$, $2n - 4$ columns if $p \neq q$. Furthermore, every row of $X(p, q)$ is either a zero row—if it corresponds to a tour containing the arc $(q, p)$—or else has exactly two 1's, one in a column of the form $(i, p)$ for some $i \in N \backslash \{p, q\}$, the other in a column of the form $(q, j)$ for some $j \in N \backslash \{p, q\}$. Let $\tilde{X}(p, q)$ be the submatrix of $X(p, q)$ consisting of those rows with exactly two 1's. Furthermore, let $H(p, q) = (V_1 \cup V_2, E)$ be the bipartite multigraph whose edge-vertex incidence matrix is $\tilde{X}(p, q)$; i.e., let $V_1$ contain a vertex for every column $(i, p)$, $i \in N \backslash \{p, q\}$, let $V_2$ contain a vertex for every column $(q, j)$, $j \in N \backslash \{p, q\}$, and let $E$ contain an edge for every row of $\tilde{X}(p, q)$. We claim that $H(p, q)$ is connected. It then follows that $H(p, q)$ contains a spanning tree as a subgraph and hence the rank of its edge-vertex incidence matrix $\tilde{X}(p, q)$, and therefore the rank of $X(p, q)$, is $|A(p, q)| - 1$, i.e., $2n - 3$ if $p = q$ and $2n - 5$ if $p \neq q$.

To prove this claim, suppose $H(p, q)$ is disconnected, and let $S$ be the vertex set of one of its components. Then any edge of $H(p, q)$ that has one of its ends in $S \cap V_1$ has the other end in $S \cap V_2$; i.e., a tour $x$ in $G$ satisfying $\alpha x = \alpha_0$ contains some arc of the form $(i, p) \in S$ if and only if it also contains some arc of the form $(q, j) \in S$. In other words, every tour $x$ satisfying $\alpha x = \alpha_0$ also satisfies

$$x(S \cap V_1) = x(S \cap V_2). \tag{17}$$

Equation (17) is linearly independent of the equality set of $P^*$, and is also not a linear combination of the latter with $\alpha x = \alpha_0$. To see this it suffices to notice that every equation of (2), as well as $\alpha x = \alpha_0$, can be weakened to an inequality satisfied by all $x \in P^*$, whereas (17) cannot be so weakened: replacing "=" by either "$\leqq$" or "$\geqq$" in (17) yields inequalities invalid for $P^*$. But then it follows that the face defined by $\alpha x = \alpha_0$ has dimension of at most dim $P^* - 2$, i.e., $\alpha x = \alpha_0$ does not define a facet of $P^*$. □

Before stating our main result, we will consider separately three cases with small $n$ and $|T|$. This will take care of the two exceptions from the main theorem and also create the basis for the induction used in proving that theorem.

The shortest odd CAT has length 5 and it uses four nodes; i.e., (16) is defined only for $|T| \geqq 5$ and $n \geqq |N[T]| \geqq 4$. The first exception occurs for $n = 5$ and $|N[T]| = 4$.

PROPOSITION 4.2. *Let $|T| = 5$; i.e., $|N[T]| = 4$. Then the inequality (16) defines a facet of $P$ and $P^*$ if $n = 4$ and if $n = 6$, but not if $n = 5$.*

*Proof.* For $n \leqq 5$, any asymmetric assignment is a tour; hence $P = P^*$. If $|T| = 5$, then $T$ is the odd CAT shown in Fig. 5; i.e., numbering the nodes of $N[T]$ clockwise starting with the neutral node, $T = \{(1, 2), (2, 1), (2, 4), (3, 2), (3, 4)\}$, and $C_1 = \phi$.

First let $n = 4$. Table 1 exhibits five affinely independent extreme tours of $G$. But $5 = n^2 - 3n + 1$, hence (16) defines a facet of $P$ and $P^*$.

Now let $n = 5$; then we are in the situation shown in Fig. 6(a). Let the four nodes of $N[T]$ be numbered as before, and let $N \backslash N[T] = \{5\}$. Then there are only ten distinct extreme tours in $G$, i.e., tours containing exactly two arcs of $T \cup C_1 = T$, namely (1, 2, 3, 4, 5), (1, 2, 4, 3, 5), (1, 2, 4, 5, 3), (1, 2, 5, 3, 4), (1, 3, 2, 4, 5), (1, 3, 4, 5, 2), (1, 4, 5, 3, 2), (1, 5, 3, 2, 4), (1, 5, 3, 4, 2), (1, 5, 4, 3, 2). But $n^2 - 3n + 1 = 11 > 10$, and so in this case (16) does not define a facet of either $P$ or $P^*$. In fact, every extreme tour with respect to (16) satisfies the equation $x_{14} + x_{31} + x_{34} + x_{43} = x_{12} + x_{21}$, which is independent of $x(T \cup C_1) = (t - 1)/2$ and the degree constraints (2).

Finally, let $n = 6$, with $N[T]$ the same as before, and $N \backslash N\{t\} = \{5, 6\}$ (see Fig.

TABLE 1
*Extreme tours for $n = |N[T]| = 4$.*

| 12 | 24 | 42 | 32 | 14 | 13 | 21 | 23 | 31 | 34 | 41 | 43 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  | 1 |  | 1 | 1 |  |
| 1 | 1 |  |  |  |  |  |  | 1 |  |  | 1 |
|  |  | 1 |  |  | 1 | 1 |  |  | 1 |  |  |
|  | 1 |  | 1 |  | 1 |  |  |  |  | 1 |  |
|  |  |  | 1 | 1 |  | 1 |  |  |  |  | 1 |

TABLE 2
*Extreme tours for* $n = 6$, $|N[T]| = 4$.

| 23 | 31 | 13 | 21 | 14 | 25 | 35 | 15 | 42 | 64 | 51 | 53 | 52 | 26 | 36 | 16 | 54 | 65 | 41 | 12 | 24 | 34 | 46 | 32 | 43 | 45 | 56 | 61 | 62 | 63 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  |  |  | 1 | 1 | 1 |  |  |
|  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |  | 1 | 1 |  |  | 1 |
|  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  | 1 |  | 1 | 1 | 1 |  |  |
|  |  | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  | 1 | 1 |  | 1 |  |
|  |  |  | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 | 1 |  |  | 1 |
|  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |  | 1 |  |  |  |  | 1 |  |  | 1 |
|  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  | 1 |  | 1 | 1 |  |  |
|  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  |  | 1 |  |  | 1 |  |  | 1 |
|  |  |  | 1 |  |  |  | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  | 1 |  |  | 1 |
|  |  |  | 1 |  |  |  | 1 |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |  | 1 |  |  |  |
|  |  |  |  |  |  | 1 |  |  |  | 1 |  |  |  |  |  |  |  |  | 1 | 1 |  | 1 |  |  |  |  |  |  | 1 |
|  | 1 |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  | 1 |  | 1 | 1 |  | 1 |  |  |  |  |  |  |  |
|  |  | 1 | 1 |  |  |  |  |  |  |  |  | 1 |  |  |  |  | 1 |  |  |  | 1 | 1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | 1 |  |  | 1 |  |  |  |  |  | 1 |  | 1 |  |  |  | 1 |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  | 1 |  |  |  | 1 |  |  | 1 |  | 1 | 1 |  |  |  | 1 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  | 1 |  | 1 | 1 |  | 1 |  |  | 1 |  |  |  |  |  |  |
|  |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 | 1 |  |  |  |  |  | 1 | 1 |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  | 1 |  | 1 |  | 1 | 1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  |  |  | 1 | 1 | 1 |  | 1 |  |  |  |  |  |  |  |  |

7(a)). Table 2 exhibits 19 extreme tours of $G$ arranged in a sequence that makes it easy to check that the first 19 columns of the table form a nonsingular matrix.

Thus the tours are affinely independent and since $19 = n^2 - 3n + 1$, (16) defines a facet of $P$ and $P^*$. □

The next proposition deals with the case $|N[T]| = 5$.

PROPOSITION 4.3. *Let* $|T| = 7$ *and* $|N[T]| = 5$. *Then the inequality* (16) *defines a facet of* $P$ *and* $P^*$ *for* $n = 5$ *and* $n = 6$.

*Proof.* If $|T| = 7$ and $|N[T]| = 5$, $T$ is of the form shown in Fig. 6(b) or 6(c). Since these are analogous, suppose we are in case 6(b), and the nodes are numbered counterclockwise, starting with the neutral node. Then $T = \{(1, 2), (2, 1), (2, 3), (3, 4), (4, 3), (5, 4), (5, 2)\}$ and $C_1 = \{(2, 4), (5, 3)\}$.

First let $n = 5$ $(= |N[T]|)$. Table 3 exhibits for this case $11 = n^2 - 3n + 1$ affinely independent extreme tours, hence (16) is facet defining for both $P$ and $P^*$.

Now let $T$ be as above, but $n = 6$. This is the situation shown in Fig. 7(b) and 7(c). Assume, without loss of generality, that we are in situation 7(b), with the first five nodes

TABLE 3
*Extreme tours for* $n = |N[T]| = 5$.

| 12 | 15 | 25 | 43 | 24 | 45 | 32 | 51 | 13 | 14 | 42 | 23 | 31 | 34 | 35 | 41 | 21 | 52 | 53 | 54 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  |  |  |  |  | 1 |  |  | 1 | 1 |  |  |  | 1 |
|  | 1 |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  | 1 |  | 1 |  |  |
| 1 |  | 1 |  |  |  |  |  |  |  |  |  |  | 1 |  | 1 |  |  | 1 |  |
| 1 |  | 1 | 1 |  |  |  |  |  |  |  |  | 1 |  |  |  |  |  |  | 1 |
|  | 1 |  | 1 | 1 |  |  |  |  |  |  |  | 1 |  |  |  |  | 1 |  |  |
| 1 |  |  |  | 1 | 1 |  |  |  |  |  |  | 1 |  |  |  |  |  | 1 |  |
|  | 1 |  | 1 |  |  | 1 |  |  |  |  |  |  |  |  |  | 1 |  |  | 1 |
| 1 |  |  |  |  | 1 |  | 1 |  |  |  | 1 |  | 1 |  |  |  |  |  |  |
|  |  |  |  |  | 1 |  |  | 1 |  |  |  |  | 1 |  |  | 1 | 1 |  |  |
|  |  |  | 1 |  |  |  |  |  | 1 |  |  |  |  | 1 |  | 1 | 1 |  |  |
|  | 1 |  |  |  |  |  |  |  |  | 1 |  |  | 1 |  |  | 1 |  | 1 |  |

belonging to $N[T]$, numbered as before and with $N\backslash N[T] = \{6\}$. Table 4 displays $19 = n^2 - 3n + 1$ affinely independent extreme tours of $G$. Thus (16) defines a facet of $P$ and $P^*$ in this case also. □

Finally, for $|N[T]| = 6$ there is also an exception from our main theorem, namely for the CAT of Fig. 7(f) and $n = 6$. We will denote by $T^*$ the odd CAT of Figure 7(f); i.e., $T^*$ consists of three neutral 2-cycles whose non-neutral nodes are joined in a 3-cycle.

PROPOSITION 4.4. *If* $n = 6$ *and* $T = T^*$, *then* (16) *does not define a facet of* $P$ *or* $P^*$.

*Proof.* Note first that an asymmetric assignment in $G$ is either a tour or the union of two disjoint 3-cycles. Since any pair of disjoint 3-cycles contains at most three arcs of $T^*$, and since $(|T^*| - 1)/2 = 4$, every asymmetric assignment that satisfies (16) with equality is a tour. So (in this case) (16) is facet-defining either for both of $P$ and $P^*$, or for none of them.

Since a tour has six arcs, in order to contain four arcs of $T^*$ it must contain either a 4-path in $T^*$, or else a 3-path and a (node-)disjoint arc. Altogether there are six distinct 4-paths and six distinct 3-paths in $T^*$, and each of the 3-paths can be combined with two distinct arcs (node-)disjoint from the given 3-path. This gives a total of $6 + 2 \times 6 = 18$ distinct extreme tours. But for (16) to be facet-defining, $G$ would have to contain $n^2 - 3n + 1 = 19$ affinely independent tours. □

We are now ready to state our main result.

THEOREM 4.5. *Let* $G = (N, A)$ *be the complete directed graph on n nodes. Let* $T$ *be an odd* CAT, $T \neq T^*$, *and let* $C_1$ *be the set of its chords of type* 1. *Then for all* $n \geqq 6$ *the inequality*

$$(16) \qquad x(T \cup C_1) \leqq (t-1)/2$$

*defines a facet of* $P$ *and* $P^*$.

*Proof.* We will exhibit $\dim P^* = n^2 - 3n + 1$ affinely independent tours, extreme with respect to (16). We will do this by using induction on $n$ in a vein similar to that of a recent proof of Fischetti [1988] for another class of inequalities, but we will rely heavily on Theorem 4.1 above.

TABLE 4
*Extreme tours for* $n = 6$, $|N[T]| = 5$.

| 12 | 52 | 26 | 43 | 24 | 46 | 32 | 51 | 13 | 14 | 42 | 62 | 61 | 25 | 45 | 35 | 56 | 63 | 64 | 23 | 31 | 34 | 41 | 21 | 53 | 54 | 16 | 36 | 65 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | 1 | | 1 | 1 | |
| | 1 | | | | | | | | | | | | | | | | | | 1 | | 1 | 1 | | | | 1 | | 1 | |
| 1 | | 1 | | | | | | | | | | | | | | | | | | | 1 | 1 | | 1 | | | | 1 | |
| 1 | | 1 | 1 | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | 1 | |
| | 1 | | 1 | 1 | | | | | | | | | | | | | | | | 1 | | | | | | 1 | | 1 | |
| 1 | | | | 1 | 1 | | | | | | | | | | | | | | | 1 | | | | 1 | | | | 1 | |
| | | | 1 | | | 1 | | | | | | | | | | | | | | | | | 1 | | 1 | 1 | | 1 | |
| 1 | | | | | 1 | | 1 | | | | | | | | | | | | 1 | | 1 | | | | | | | 1 | |
| | 1 | | | | 1 | | | 1 | | | | | | | | | | | | | 1 | | 1 | | | | | 1 | |
| | 1 | | 1 | | | | | | 1 | | | | | | | | | | | | | | 1 | | | | 1 | 1 | |
| | | | | | | | | | | 1 | | | | | | | | | | | 1 | | 1 | 1 | | 1 | | 1 | |
| | | | 1 | | | | | | | | 1 | | | | | | | | | | | | 1 | | 1 | | 1 | | 1 |
| | 1 | | 1 | 1 | | | | | | | | 1 | | | | | | | | | | | | | | | 1 | | 1 |
| 1 | | | 1 | | | | | | | | | 1 | 1 | | | | | | | | | | | | 1 | | 1 | | |
| 1 | | | | 1 | | | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | |
| 1 | | | | | 1 | | | | | | | 1 | | | 1 | | | | 1 | | | | | | 1 | | | | |
| 1 | | | | | | | | | | | | 1 | | 1 | | 1 | | | 1 | | 1 | | | | | | | | |
| | 1 | | | | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | 1 | | | |
| | 1 | | 1 | | | | | | | | | | | | 1 | | | 1 | | | | | 1 | | | 1 | | | |

For the purposes of induction, we will have to work with, besides $G$, the complete digraphs $G'$ and $G''$ obtained from $G$ by deleting one and two nodes, respectively. We will denote by $P^*(G)$, $P^*(G')$ and $P^*(G'')$ the corresponding ATS polytopes. Also, for any $k \in N$, $S \subseteq N - \{k\}$, we will denote $(k, S) := \{(k, j) : j \in S\}$, and $(S, k) := \{(i, k) : i \in S\}$.

There are five cases to be considered. In each of these cases we assume (based on Propositions 4.2 and 4.3) that the statement in the theorem holds for $n = |N[T]| = 5$ (for $n = 6$ if $|N[T]| = 4$), and we let $n \geqq \max\{6, n_0\}$ where $n_0$ is the smallest value of $n$ for which the case in question can arise. In the first four cases we do not assume anything about $N \backslash N[T]$, which may or may not be empty: our induction arguments affect only the nodes of $N[T]$. In the last case, the induction involves a node in $N \backslash N[T]$, which is assumed to be nonempty.

*Case* 1. $T$ has a 2-cycle whose two nodes are both sources or both sinks. Since these two situations are analogous, we deal explicitly only with the first one. Without loss of generality, let $n = |N|$ and $p = n - 1$, let $n$ and $p$ be the nodes of the 2-cycle, and let $(n, r)$ be the arc of $T$ (other than $(n, p)$) incident from $n$ (see Fig. 8(a)).

Define $G' := G - \{n\} = (N', A')$ and

$$T' := (T \backslash \{(p,n),(n,p),(n,r)\}) \cup \{(p,r)\}.$$

Then $T'$ is an odd CAT in $G'$ with $t' = |T'| = t - 2$, and its set of chords of type 1 is

$$C_1' = C_1 \backslash \{(n, U),(p,r)\},$$

where $U$ is the set of sinks of $T$. By the induction hypothesis, the inequality

$$(16') \qquad x(T' \cup C_1') \leqq (t-3)/2$$

defines a facet of $P^*(G')$. Hence there exists a set of $q := p^2 - 3p + 1$ affinely independent extreme tours in $G'$. Let $y^i$, $i = 1, \cdots, q$, be these tours. From each $y^i$ we construct a tour $x^i$ in $G$ by inserting $n$ after $p$; i.e., if $j_i$ is the successor of $p$ in tour $y^i$, we replace arc $(p, j_i)$ with the pair of arcs $(p, n)$, $(n, j_i)$. Since $(p, n) \in T$, and since $(n, j_i) \in T \cup C_1$ if and only if $(p, j_i) \in T' \cup C_1'$, it follows that $x^i$ always contains one more arc of $T \cup C_1$ than $y^i$ contains arcs of $T' \cup C_1'$; therefore, as $t = t' + 1$, $x^i$ is extreme for each $i$.

Also, the $x^i$, $i = 1, \cdots, q$, are affinely independent. To see this it suffices to notice that for $i = 1, \cdots, q$,

$$(18) \qquad x_{kl}^i = \begin{cases} y_{kl}^i & k = 1, \cdots, p-1, l = 1, \cdots, p, l \neq k \\ 0 & k = p, l = 1, \cdots, p-1; \\ & k = 1, \cdots, p-1, l = n; \text{ and } k = n, l = p \\ y_{pl}^i & k = n, l = 1, \cdots, p-1 \\ 1 & k = p, l = n \end{cases}$$

and so the affine dependence of the $x^i$ would imply that of the $y^i$, a contradiction.

We need $n^2 - 3n + 1 - q = 2p - 2$ additional extreme tours. Let $Y$ and $X_1$ be the matrices whose rows are the $q$ vectors $y^i$ and $x^i$, $i = 1, \cdots, q$, respectively. Notice that there are $2p - 1$ "free" arcs, i.e., arcs not contained in any of the $q$ tours $x^i$ constructed so far. In other words, the matrix $X_1$ has $2p - 1$ zero columns, specified in (18).

Consider now another $q$ extreme tours in $G$ constructed from the same extreme tours $y^i$ in $G'$ by inserting $n$ before rather than after $p$. The construction is perfectly

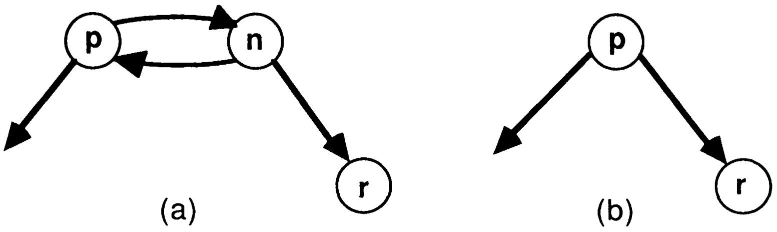


FIG. 8

analogous to the above one, and results in $q$ tours $x^i$, $i = q+1, \cdots, 2q$, such that

$$(19)\qquad x^i_{kl} = \begin{cases} y^i_{kl} & k=1,\cdots,p,\ l=1,\cdots,p-1,\ l\neq k \\ 0 & k=1,\cdots,p-1,\ l=p; \\ & k=n,\ l=1,\cdots,p-1;\text{ and } k=p,\ l=n \\ y^i_{kp} & k=1,\cdots,p-1,\ l=n \\ 1 & k=n,\ l=p. \end{cases}$$

Let the matrix whose rows are these tours $x^i$, $i = q+1, \cdots, 2q$, be $X_2$, and let $X$ be the matrix whose rows are the $2q$ rows of $X_1$ and $X_2$. By permuting the columns of $X$ appropriately, it can be brought to the form

$$X = \begin{pmatrix} X_{11} & 0 \\ X_{21} & X_{22} \end{pmatrix}$$

where 0 is the $q \times (2p-1)$ matrix containing the zero columns of $X_1$, $X_{11}$ is the rest of $X_1$, and $X_{21}$, $X_{22}$ form the corresponding partition of $X_2$.

Consider the submatrix $X'_{22}$ of $X_{22}$ obtained by removing column $(n, p)$. It consists of the $2p-2$ columns of $X_2$ of the form $(k, n)$, $k \in N' \setminus \{p\}$, and $(p, l)$, $l \in N' \setminus \{p\}$. From (19), the columns of $X'_{22}$ corresponding to arcs $(p, l)$, $l \in N' \setminus \{p\}$, are the same as the corresponding columns of $Y$; and the columns corresponding to arcs $(k, n)$, $k \in N' \setminus \{p\}$, are copies of the columns of $Y$ corresponding to arcs $(k, p)$, $k \in N' \setminus \{p\}$. Let $Y_0$ be the submatrix of $Y$ made up of these $2p-2$ columns. From Theorem 4.1, the rank of $Y_0$ is $2p-3$, since by the induction hypothesis (16′) defines a facet of $P^*(G')$. Hence $X'_{22}$, which is a copy of $Y_0$, also has rank $2p-3$. Since $X_{11}$ has rank $q = p^2 - 3p + 1$, this implies that the rank of $X$ is $\geqq q + (2p-3)$.

Now consider any extreme tour $x^*$ in $G$, which contains neither of the two arcs $(p, n)$, $(n, p)$. Such $x^*$ can be obtained from any extreme tour $y^*$ containing an arc $(v, w)$ with $\deg^-_{T'}(w) = 2$ and $\deg^+_{T'}(v) < 2$, by inserting $n$ into $(v, w)$. Since by hypothesis every arc of $G'$ is contained in at least one extreme tour, such $y^*$ obviously exists. Then $(v, w) \notin T' \cup C'_1$ and $(v, n) \notin T \cup C_1$, but $(n, w) \in T \cup C_1$ (since $n$ is a source and $w$ is a sink); and thus $x^*(T \cup C_1) = y^*(T' \cup C'_1) + 1$, i.e., $x^*$ is extreme. We claim that $x^*$ is affinely independent of the $2q$ extreme tours that form the rows of $X$. For suppose not; then there exist scalars $\lambda_i$, $i = 1, \cdots, 2q$, such that $\sum(\lambda_i : i = 1, \cdots, 2q) = 1$ and

$$\sum((x^i_{pn} + x^i_{np})\lambda_i : i = 1, \cdots, 2q) = x^*_{pn} + x^*_{np}$$

or, as $x^i_{pn} + x^i_{np} = 1$, $i = 1, \cdots, 2q$ and $x^*_{pn} + x^*_{np} = 0$,

$$\sum(\lambda_i : i = 1, \cdots, 2q) = 0,$$

a contradiction. Thus the matrix obtained from $X$ by adding $x^*$ as a row, has rank $q + (2p-3)+1 = n^2 - 3n + 1$. This completes the proof of Case 1.

*Case* 2. $T$ has a sequence of the form $\{(r, n), (p, n), (p, q), (n, q), (n, s)\}$, where $\deg^+_T(p) = \deg^+_T(n) = \deg^-_T(n) = \deg^-_T(q) = 2$ and $\deg^+_T(q) = \deg^-_T(p) = 0$ (see Fig. 9). Let $n = |N|$, $p = n-1$, and define $G' := G - \{n\} = (N', A')$, with $T' := (T\setminus\{(r, n), (p, n), (n, q), (n, s)\}) \cup \{(r, q), (p, s)\}$. Then $T'$ is an odd CAT with $t' = |T'| = t - 2$, and its set of chords of type 1 is

$$C'_1 = C_1\setminus\{(S,n),(n,U),(r,q),(p,s)\}$$

where $S$ and $U$ are the sources and sinks of $T$. Again by the induction hypothesis, the inequality (16′) defines a facet of $P^*(G')$. Let $y^i$, $i = 1, \cdots, q = p^2 - 3p + 1$, be affinely independent extreme tours in $G'$, whose existence follows from the induction hypothesis. For $i = 1, \cdots, q$, let $x^i$ be the tour in $G$ obtained from $y^i$ by inserting $n$ after $p$. By the same argument as in Case 1, the $q$ tours $x^i$ are affinely independent. Furthermore, they are extreme since $(T' \cup C'_1) \subset (T \cup C_1)$ and the insertion of $n$ replaces an arc of $y^i$ in $T' \cup C'_1$ with two arcs of $x^i$ in $T \cup C_1$, or an arc of $y^i$ not in $T' \cup C'_1$ by two arcs of $x^i$, one in $T \cup C_1$ and one not in $T \cup C_1$.

Another set of $q$ affinely independent extreme tours $x^i$, $i = q+1, \cdots, 2q$, can be obtained by inserting $n$ before $q$. Their mutual independence and extremality follows by the same argument as for the first set of $q$ tours. Among the tours of this second set, we will identify $2p-5$ that, together with the first $q$ tours, form an affinely independent set. For this purpose, we first remove from the second set all those tours identical to tours of the first set. These are precisely the tours $x^i$ obtained from tours $y^i$ of $G'$ containing the arc $(p, q)$. Next, we denote again by $X_1$ and $X_2$ the matrices whose rows are the two sets of tours of $G$ generated by inserting $n$ after $p$ and before $q$, respectively (with $X_2$ containing only rows distinct from those of $X_1$). After appropriate column permutations we then have

$$X = \begin{pmatrix} X_{11} & 0 \\ X_{21} & X_{22} \end{pmatrix},$$

where $(X_{11}, 0)$ and $(X_{21}, X_{22})$ are obtained by the same column permutations from $X_1$ and $X_2$, respectively, and where the $2p-3$ columns of 0 and $X_{22}$ are of the form $(p, j), j \in N'\setminus\{p\}$, $(i, n)$, $i \in N'\setminus\{p\}$, and $(n, p)$. We note that no tour of the second set contains either of the three arcs $(p, q)$, $(q, n)$, or $(n, p)$, so these three columns are 0 also in $X_{22}$. Then by arguments analogous to those of Case 1, $X_{22}$ can be shown to have rank $\geqq 2p-5$.

We need three more tours. For the first one we choose any extreme tour $y^i$ of $G'$ that contains the arc $(p, q)$ but not the arc $(r, p)$, and extend it to an extreme tour $x^i$ of

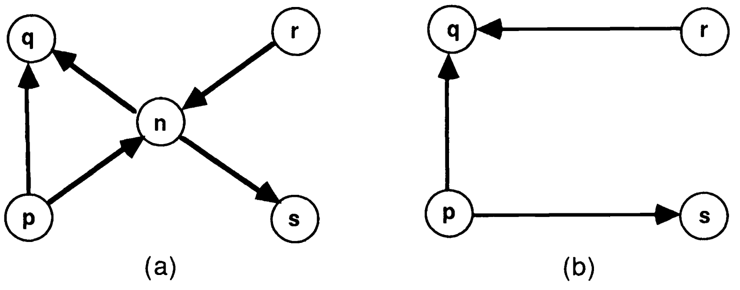


FIG. 9

$G$ by inserting $n$ after $r$. Such $y^i$ exists. For let $\hat{y}$ be an extreme tour of $G'$ that contains $(p, q)$. If $\hat{y}$ does not contain $(r, p)$, let $y^i := \hat{y}$; otherwise let $y^i$ be the tour of $G'$ obtained from $\hat{y}$ by deleting the arcs $(r, p)$, $(q, t)$ and $(u, v)$, where $t$ is the successor of $q$ in $\hat{y}$ and $(u, v) \notin T' \cup C'_1$ is any arc contained in the segment of the tour $\hat{y}$ between $t$ and $r$. Then $y^i(T' \cup C'_1) = \hat{y}(T' \cup C'_1)$, i.e., $y^i$ is extreme. To obtain the second and third tours we choose extreme tours of $G'$ that do not contain $(p, q)$, but contain an arc $(q, j)$, $j \in U\setminus\{q, n\}$ (the second tour), or an arc $(h, p)$, $h \in S\setminus\{p, n\}$ (the third tour). Such tours exist for the same reasons that $y^i$ exists. We then extend these tours of $G'$ to extreme tours of $G$ by inserting $n$ after $q$, i.e., into the arc $(q, j)$ (for the second tour), or before $p$, i.e., into the arc $(h, p)$ (for the third tour). The three extreme tours obtained this way are affinely independent of each other and of all the remaining tours, because each one has an arc not contained in any other tour. This raises the number of tours constructed to $n^2 - 3n + 1$ and completes the proof of Case 2.

*Case* 3. $T$ has two neutral 2-cycles joined by an arc, and $T \neq T^*$. Let $k$, $l$ and $p$, $n$ be the two pairs of nodes of the 2-cycles, with $(k, p)$ the arc joining them (see Fig. 10).

Since $T \neq T^*$, the smallest odd CAT containing this configuration has 8 nodes; thus we assume for this case that $n \geqq |N[T]| \geqq 8$. Again, without loss of generality, let $n = |N|$ and $p = n - 1$, and define $G' := G - \{n\} = (N', A')$ and

$$T' := (T\setminus\{(p,n),(n,p),(k,p)\}) \cup \{(p,l)\}.$$

Then $T'$ is an odd CAT in $G'$ with $t' = |T'| = t - 2$, and its set of chords of type 1 is

$$C'_1 = (C_1\setminus\{(k,U),(S,p)\}) \cup \{(S\setminus\{p\},l)\}$$

where $S$ and $U$ are the sets of sources and sinks, respectively, of $T$. As in Case 1, by the induction hypothesis the inequality (16′) defines a facet of $P^*(G')$. Hence there exists a set of $q := p^2 - 3p + 1$ affinely independent extreme tours $y^i$ in $G'$. We will require in addition each of these tours to satisfy the inequality

$$(20) \qquad y^i(S\setminus\{k\},l) \leqq y^i(S\setminus\{p\},p) + y^i(k, U\setminus\{k,p\}).$$

We will show that there always exists a set of $q$ such tours, with the additional property that $2p - 2$ among these tours also satisfy the inequality

$$(21) \qquad y^i(S\setminus\{k\},l) + y^i(p, U\setminus\{p\}) \leqq y^i(S\setminus\{p\},p) + y^i(k, U\setminus\{k,p\}).$$

We claim that

(i) any extreme tour $y^i$ in $G'$ that satisfies (20) can be extended to an extreme tour in $G$ by inserting $n$ before $p$; and

(ii) any extreme tour $y^i$ in $G'$ that satisfies (21) can be extended to an extreme tour in $G$ by inserting $n$ after $p$.

Now let $y^i$ be any extreme tour in $G'$, and let $j_1$, $j_2$, $j_3$ and $j_4$ be the predecessor of $l$, the successor of $k$, the predecessor $p$ and the successor of $p$, respectively, in the tour $y^i$.

To prove claim (i), let $x^i$ be the tour in $G$ obtained from $y^i$ by inserting $n$ before $p$, and suppose $x^i$ is not extreme. Then $x^i$ contains no more arcs of $T \cup C_1$ than $y^i$ contains arcs of $T' \cup C'_1$, in spite of the fact that $(n, p) \in T \cup C_1\setminus(T' \cup C'_1)$. This implies that $y^i$ contains more arcs of $T' \cup C'_1$ than of $T \cup C_1$, i.e., $j_1 \in S\setminus\{k\}$, $j_2 \notin U\setminus\{k, p\}$, and $j_3 \notin S\setminus\{p\}$. Thus the right-hand side of (20) is 0 while its left-hand side is 1, i.e., $y^i$ violates (20).

To prove claim (ii), if $x^i$ is obtained from $y^i$ by inserting $n$ after $p$ and $x^i$ is not extreme, then either $y^i$ contains more arcs of $T' \cup C'_1$ than of $T \cup C_1$, i.e., $j_1 \in S\setminus\{k\}$ and $j_2 \notin U\setminus\{k, p\}$, $j_3 \notin S\setminus\{p\}$, or else the arc $(p, j_4)$ of $y^i$ (which is not an arc of $x^i$)

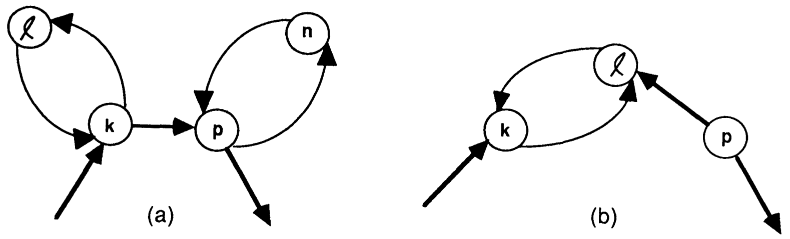


FIG. 10

is contained in $T \cup C_1$, i.e., $j_4 \in U \setminus \{p\}$, but $j_2 \notin U \setminus \{k, p\}$ and $j_3 \notin S \setminus \{p\}$. In either case the right-hand side of (21) is 0, while the left-hand side is $\geqq 1$, i.e., $y^i$ violates (21).

We claim that there always exist $q := p^2 - 3p + 1$ affinely independent extreme tours in $G'$ that satisfy condition (29). To prove the claim, we note that nodes $k$, $l$ and $p$ of $G'$ induce in $T'$ the configuration treated under Case 1. We can then obtain the $p^2 - 3p + 1$ affinely independent extreme tours $y^i$ of $G'$ by the technique used in the proof of Case 1, slightly amended to result in tours that satisfy (20). Let $G'' := G' - \{l\}$, and let nodes $k$ and $l$ of $G'$ play the role of nodes $p$ and $n$ of $G$ in the proof of Case 1. Then of the $p^2 - 3p + 1$ extreme tours $y^i$ of $G'$, those $(p-1)^2 - 3(p-1) + 1$ obtained by inserting node $l$ *after* $k$ into the corresponding extreme tours of $G''$ cannot contain any arc of the set $(S \setminus \{k\}, l)$, since they contain $(k, l)$; and so these tours satisfy (20). Of the $(p-1)^2 - 3(p-1) + 1$ tours (constructed by the procedure of Case 1) that contain $(l, k)$, we choose only those that satisfy (20); there are among the latter $2p - 5$ that are affinely independent of each other and of all the tours containing $(k, l)$, as we will presently show. This provides a total of $(p-1)^2 - 3(p-1) + 1 + (2p - 5) = p^2 - 3p$ affinely independent extreme tours. Finally, for a last extreme tour we choose any one that satisfies (20) and contains neither $(k, l)$, nor $(l, k)$.

Now to prove that the tours containing arc $(l, k)$ add $2p - 5$ to the rank of the incidence matrix of tours versus arcs, let $Y_1$ and $Y_2$ be the matrices whose rows are all the tours of $G'$ constructed from tours in $G''$ by inserting $l$ after $k$ and before $k$, respectively, and let $Y(l, k)$ be the submatrix of $Y_2$ consisting of all columns of the form $(i, l)$, $i \in N' \setminus \{l, k\}$ and $(k, j)$, $j \in N' \setminus \{l, k\}$. From Theorem 4.1, the rank of $Y(l, k)$ is $2p - 5$. Also, the submatrix of $Y_1$ corresponding to $Y(l, k)$ is a zero matrix. So all we need to show is that removing those rows of $Y(l, k)$ corresponding to tours that violate (20) does not reduce the rank of $Y(l, k)$.

We prove this by showing that the bipartite multigraph $H(l, k)$ whose edge-vertex incidence matrix is $Y(l, k)$ (see the proof of Theorem 4.1) remains connected after the removal of all edges corresponding to tours that violate (20). Let $H^*(l, k)$ be the subgraph of $H(l, k)$ obtained by the removal of these edges, and let $(i, l)$, $(k, j)$ be any pair of arcs of $G'$ corresponding to an edge of $H(l, k)$ that has been removed. We will show that $(i, l)$ is connected to $(k, j)$ in $H^*(l, k)$.

Since the tour containing the sequence $D := \{(i, l), (l, k), (k, j)\}$ violates (20), it follows that $i \in S \setminus \{k\}$ and $j \notin U \setminus \{k, p\}$. We exhibit three sequences of arcs, $D_1$, $D_2$, $D_3$, each of which can be completed to an extreme tour in $G'$ that satisfies (20) and hence corresponds to an edge of $H^*(l, k)$. These three edges then form a path in $H^*(l, k)$ that connects $(i, l)$ to $(k, j)$. The three sequences are:

$$D_1 := \{(i,l),(l,k),(k,h)\} \text{ for some } h \in U \setminus \{k,p,i\};$$

$$D_2 := \{(p,l),(l,k),(k,h)\}$$

$$D_3 := \{(i,p),(p,l),(l,k),(k,j)\}.$$

The existence of $h$ follows from $|N[T]| \geqq 8$. Since $D_1$ and $D_2$ contain $(k, h) \in (k, U \setminus \{k, p\})$, while $D_3$ contains $(i, p) \in (S \setminus \{p\}, p)$, any tour in $G'$ that contains $D_1$, $D_2$ or $D_3$ satisfies (20). Furthermore, since each $D_i$, $i = 1, 2, 3$, has at least half of its arcs in $T' \cup C'_1$, it can obviously be completed to an extreme tour in $G'$, i.e., one with $(|T'| - 1)/2$ arcs in $T' \cup C'_1$. Finally, the path in $H^*(l, k)$ that connects $(i, l)$ to $(k, j)$ consists of the three edges $((i, l), (k, h))$, $((k, h), (p, l))$, $((p, l), (k, j))$ corresponding to the three extreme tours containing $D_1$, $D_2$ and $D_3$, respectively. This completes the proof of our claim that $G'$ has $p^2 - 3p + 1$ affinely independent extreme tours that satisfy (20).

Next we claim that among the $q = p^2 - 3p + 1$ affinely independent extreme tours in $G'$ that satisfy (20), there are $2p - 2$ that also satisfy (21). Clearly, if a tour $y^i$ satisfies (20) but violates (21), and if $h$ and $j$ are the predecessor and successor, respectively, of $p$ in $y^i$, then $h \notin S \setminus \{p\}$ and $j \in U \setminus \{p\}$. Let $y$ be the matrix whose rows are the $q$ affinely independent extreme tours $y^i$, and let $Y(p)$ be the submatrix of $Y$ consisting of the $2p - 2$ columns of the form $(h, p)$, $h \in N \setminus \{p\}$, and $(p, j)$, $j \in N \setminus \{p\}$. From Theorem 4.1, the rank of $Y(p)$ is $2p - 3$. We will show that removing all the rows of $Y(p)$ corresponding to tours such that $h \notin S \setminus \{p\}$ and $j \in U \setminus \{p\}$, where $h$ and $j$ are the predecessor and successor of $p$ in the given tour, results in a matrix $Y^*(p)$ of the same rank $2p - 3$. But then $Y^*(p)$ has at least $2p - 2$ rows, which proves that $2p - 2$ of the tours $y^i$ that satisfy (20) also satisfy (21).

To prove that $Y^*(p)$ has rank $2p - 3$, we will show that it is the incidence matrix of a connected bipartite graph on $2p - 2$ vertices. Let $H(p) = (V_1 \cup V_2, E)$ be the bipartite multigraph whose edge-vertex incidence matrix is $Y(p)$, and let $H^*(p) = (V_1 \cup V_2, E^*)$ be the subgraph of $H(p)$ that is the incidence matrix of $Y^*(p)$. $H(p)$ is connected (see the proof of Theorem 4.1). $H^*(p)$, like $H(p)$ has $2p - 2$ vertices. To show that $H^*(p)$ is connected, let $(h, p)$, $(p, j)$ be any pair of arcs of $G'$ (vertices of $H^*(p)$) corresponding to an edge in $E \setminus E^*$. Then $h \notin S \setminus \{p\}, j \in U \setminus \{p\}$. We will show that $(h, p)$ is connected to $(p, j)$ in $H^*(p)$. For this purpose we exhibit three sequences of arcs, $F_1$, $F_2$, and $F_3$, each of which can be completed to an extreme tour in $G'$ satisfying (21) and hence corresponding to an edge of $H^*(p)$, such that the three edges form a path in $H^*(p)$ connecting $(h, p)$ to $(p, j)$. Consider the sequences

$$F_1 := \{(h,p),(p,l),(l,k),(k,r)\} \quad \text{for some } r \in U \setminus \{k,p,h\}$$

$$F_2 := \{(m,p),(p,l),(l,k)\} \quad \text{for some } m \in S \setminus \{p\},$$

$$F_3 := \{(m,p),(p,j),(j,l)\}.$$

Again, the existence of $r$ follows from $|N[T]| \geqq 8$. Each of $F_1$, $F_2$, and $F_3$ has at least half of its elements in $T' \cup C'_1$ and therefore can be completed to an extreme tour in $G'$. Furthermore, any such extreme tour satisfies (21) since for any tour containing $F_1$, $F_2$ or $F_3$, the left-hand side of (21) is equal to 1 while the right-hand side is greater than or equal to 1 (equal to 1 in the case of $F_1$). Finally, the path in $H^*(p)$ that connects $(h, p)$ to $(p, j)$ consists of the three edges $((h, p), (p, l))$, $((p, l), (m, p))$, $((m, p), (p, j))$.

This proves the claim that $2p - 2$ of the $q$ affinely independent tours of $G'$ satisfying (20) also satisfy (21).

We now return to the construction of $n^2 - 3n + 1$ affinely independent extreme tours in $G$. A first set of $q = p^2 - 3p + 1$ tours is obtained by inserting $n$ before $p$ into each of the $q$ tours $y^i$ of $G'$ postulated above. These tours, $x^i$, $i = 1, \cdots, q$, are affinely independent by the same argument as in Case 1, and they are extreme because every tour $y^i$ satisfies (20).

The next set of tours $x^i$ is obtained by inserting $n$ after $p$ into each of the $2p-3$ affinely independent extreme tours $y^i$ of $G'$ that satisfy (21). The resulting tours are then affinely independent and extreme. To show that they are also affinely independent of the $q$ tours $x^i$ obtained by inserting $n$ before $p$, let $X_1$ and $X_2$ be the matrices whose rows are the tours $x^i$ obtained by inserting $n$ before $p$ and after $p$, respectively, and let $Y_1$, $Y_2$ be the corresponding matrices whose rows are the tours $y^i$ in $G'$ from which the $x^i$ were obtained. Then the matrix whose rows are the tours $x^i$ in $G$ can be brought by appropriate column permutations to the form

$$X=\begin{pmatrix} X_{11} & 0 \\ X_{21} & X_{22} \end{pmatrix},$$

where $(X_{11}, 0)$ and $(X_{21}, X_{22})$ are obtained by applying the column permutations to $X_1$ and $X_2$, respectively, and where the submatrices 0 and $X_{22}$ have $2p-1$ columns, namely, $(p,j), j\in N\backslash\{p,n\}$, $(i,n)$, $i\in N\backslash\{p,n\}$, and $(p,n)$. Let $X'_{22}$ be obtained from $X_{22}$ by removing column $(p,n)$; then $X'_{22}$ is a copy of the submatrix $Y_2(p)$ of $Y_2$, whose $2p-2$ columns are $(p,j), j\in N'\backslash\{p\}$, and $(i,p)$, $i\in N'\backslash\{p\}$. Since $Y_2(p)$ was shown to have rank $2p-3$, $X'_{22}$ has the same rank.

Finally, a last tour $x^*$, not containing either of the arcs $(n,p)$, $(p,n)$, is added to the above $q+2n-3$ tours to form a set of $n^2-3n+2$ affinely independent extreme tours in $G$. Such a tour can be constructed by inserting $n$ into any arc $(u,v)\notin T'\cup C'_1\cup\{(k,p)\}$ of an extreme tour $y^*$ of $G'$ that contains $(k,p)$ but not $(p,l)$. Such $y^*$ exists. For let $\hat{y}$ be an extreme tour of $G'$ that contains $(k,p)$. If $\hat{y}$ does not contain $(p,l)$, let $y^*:=\hat{y}$; otherwise let $y^*$ be the tour of $G'$ obtained from $\hat{y}$ by deleting the arcs $(p,l)$, $(l,m)$ and $(h,k)$, where $m$ is the successor of $l$ and $h$ is the predecessor of $k$ in $\hat{y}$, and inserting arcs $(h,l)$, $(l,k)$ and $(p,m)$. Then $(p,l)\in T'\cup C'_1$, $(l,m)\notin T'\cup C'_1$ (since $l$ is a sink) and $(h,k)$ may or may not belong to $T'\cup C'_1$. On the other hand, $(l,k)\in T'\cup C'_1$; and $(h,l)\in T'\cup C'_1$ if and only if $(h,k)\in T'\cup C'_1$; thus $y^*(T'\cup C')\geqq\hat{y}(T'\cup C')$, i.e., $y^*$ is extreme.

This completes the proof for Case 3.

*Case* 4. $T$ does not contain either of the configurations treated under Cases 1, 2, and 3, but it has two adjacent nodes, say $p$ and $n$, such that $\deg_T^+(p)=\deg_T^-(n)=0$ and $\deg_T^+(n)=\deg_T^-(p)=2$ (see Fig. 11). Let $r$ and $s$ be the two nodes adjacent to $p$ and $n$, respectively, (other than $p$ and $n$).

Let $G''=(N'',A'')$ be the graph obtained from $G$ by deleting nodes $p$ and $n$, i.e., $G'':=G-\{p,n\}$. Define

$$T''':=(T\backslash\{(r,p),(n,p),(n,s)\})\cup\{(r,s)\}.$$

Then the set of chords of type 1 of $T'''$ is

$$C''_1=C_1\backslash\{(S,p),(n,U),(r,s)\},$$

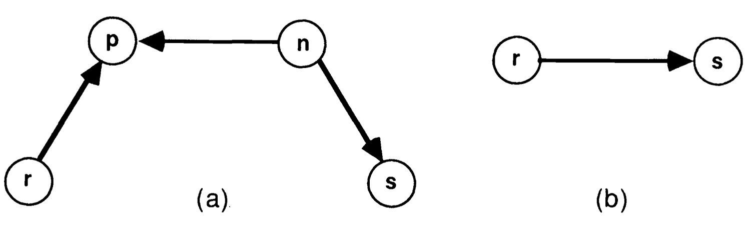


FIG. 11

where again $S$ and $U$ are the source and sink sets, respectively, of $T$. We have $|T''| = |T| - 2$ and by the induction hypothesis the inequality

$$(16'') \qquad x(T'' \cup C_1'') \leqq (t-3)/2$$

defines a facet of $P^*(G'')$.

Since (16″) defines a facet of $P^*(G'')$, there exists a set of $q := (n-2)^2 - 3(n-2) + 1 = n^2 - 7n + 11$ affinely independent extreme tours (with respect to (16″)) in $G''$. Let $y^i$, $i = 1, \cdots, q$ be the tours of such a set. We extend each of these to a tour $x^i$ in $G$ by inserting the sequence $\{p, (p, n), n\}$ after some fixed node $k$ such that $\deg^+_{T''}(k) = 2$ and $\deg^-_{T''}(k) = 0$. Such a node exists by the assumption that none of the Cases 1, 2, or 3 holds. It is easy to check, along the lines discussed in Case 1, that the resulting tours $x^i$, $i = 1, \cdots, q$, are affinely independent. Furthermore, since we always have $(k, p) \in T \cup C_1$, and since $(n, l) \in T \cup C_1$ if and only if $(k, l) \in T'' \cup C_1''$ (where $l$ is the successor of $k$ in $y^i$), each tour $x^i$ has one more arc in $T \cup C_1$ than the corresponding tour $y^i$ has in $T'' \cup C_1''$, i.e., the tours $x^i$ are extreme.

Next we generate a second set of $q$ tours $x^i$, $i = q + 1, \cdots, 2q$, by inserting the sequence $\{n, (n, p), p\}$ *before* the same node $k$ used above. Since $(n, p) \in T \cup C_1$, again each tour $x^i$ in this set has one more arc in $T \cup C_1$ than the corresponding tour $y^i$ has in $T'' \cup C_1''$ and is therefore extreme. Furthermore, the tours $x^i$ are easily seen to be affinely independent of each other, and we can use Theorem 4.1 to show that they add $2n - 7$ to the rank of the matrix $X$ whose rows are the vectors $x^i$. Indeed, if $X_1$ and $X_2$ are the matrices whose rows are the first $q$ and the second $q$ vectors $x^i$, the $n - 3$ columns of $X_1$ of the form $(k, j)$, $j \in N \backslash \{k, p, n\}$ and the $n - 3$ columns (of $X_1$) of the form $(i, n)$, $i \in N \backslash \{k, p, n\}$, form a zero matrix, and if $X_{22}$ is the submatrix of $X_2$ whose columns correspond to those of this zero matrix, from Theorem 4.1 the rank of $X_{22}$ is one less than the number of its columns, i.e., $2n - 7$. Thus the second set of $q$ vectors $x^i$ adds $2n - 7$ to the rank of the system.

We now generate from the same set of tours $y^i$ in $G''$ $q$ more tours $x^i$ by inserting the sequence $\{n, (n, p), p\}$ after some fixed node $l$ such that $\deg^-_{T''}(l) = 2$ and $\deg^+_{T''}(l) = 0$, and still $q$ more tours $x^i$ by inserting the sequence $\{p, (p, n), n\}$ before $l$. Again the existence of such $l$ is guaranteed by the fact that $G''$ does not contain either of the configurations defining Cases 1 and 2. These third and fourth sets of tours $x^i$, $i = 2q + 1, \cdots, 3q$ and $i = 3q + 1, \cdots, 4q$ are easily seen to be extreme on the same grounds as in the case of the first two sets. As to their contribution to the rank of the system, let $X_3$ and $X_4$ be the matrices whose rows are the tours in the third and fourth sets, respectively. The $n - 4$ columns of the form $(p, j)$, $j \in N \backslash \{k, l, p, n\}$, have only 0 entries in both $X_1$ and $X_2$; the corresponding submatrix of $X_3$ has exactly one 1 in every row, and its rank is $n - 4$. Furthermore, the $n - 4$ columns of the form $(i, p)$, $i \in N \backslash \{k, l, p, n\}$, have only 0 entries in each of $X_1$, $X_2$, $X_3$, and the submatrix of $X_4$ corresponding to these columns has rank $n - 4$. Thus the third and fourth set of tours $x^i$ contribute $2n - 8$ to the rank of the system which becomes $(n^2 - 7n + 11) + (2n - 7) + (2n - 8) = n^2 - 3n - 4$.

We need five more tours. As our first tour, say $x^{*1}$, we choose an extreme tour containing $(k, n)$ and $(n, p)$, but not $(p, l)$. (The presence of $(k, n)$ and $(n, p)$ excludes $(n, k)$ and $(l, p)$). Such a tour can be constructed as follows. Let $y^*$ be any extreme tour of $G''$ in which $k$ is followed by a node, say $m$, such that $\deg^-_{T''}(m) = 0$, and let $x^*$ be the extreme tour of $G$ obtained from $y^*$ by inserting the sequence $(n, (n, p), p)$ after the node $l$ as explained earlier in this proof. Then $x^*$ contains $(n, p)$ and $(k, m)$ but not $(p, l)$. We can then replace $(k, m)$ by $(k, n)$ as follows. Notice that $(k, m) \notin T \cup C_1$ and $(l, n) \notin T \cup C_1$. Let $(u, v) \notin T \cup C_1$ be any arc of $x^*$ distinct from $(k, m)$ and $(l, n)$.

(Since $|T| \geqq 7$, $x^*$ has more than 3 arcs not contained in $T \cup C_1$.) Without loss of generality, suppose $x^*$ traverses the arcs $(k, m)$, $(l, n)$ and $(u, v)$ in that order (an analogous argument holds for a different order). Then deleting $(k, m)$, $(l, n)$ and $(u, v)$ and inserting $(k, n)$, $(l, v)$ and $(u, m)$ produces a tour $x^{*1}$ of $G$ containing $(k, n)$ and $(n, p)$ but not $(p, l)$, such that $x^{*1}(T \cup C_1) = x^*(T \cup C_1)$, i.e., extreme.

As our second extreme tour we choose $x^{*2}$ containing $(n, k)$ and $(p, n)$ but not $(l, p)$. The construction of $x^{*2}$ is analogous to that of $x^{*1}$. As our third and fourth tours we choose $x^{*3}$ containing $(l, p)$ and $(p, n)$ and $x^{*4}$ containing $(p, l)$ and $(n, p)$. Clearly, the four tours $x^{*i}$, $i = 1, \cdots, 4$ are independent from each other and from the tours constructed earlier.

Finally, our last tour $x^{*5}$ can be any extreme tour of $G$ containing neither $(n, p)$ nor $(p, n)$. The existence of such a tour is easily seen. Since every one of the tours constructed earlier contains either $(n, p)$ or $(p, n)$ while $x^{*5}$ contains neither of these arcs, $x^{*5}$ can be shown to be affinely independent of the other extreme tours by the argument used at the end of Case 1.

This completes the proof of Case 4.

*Case* 5. $n \geqq 7$ and $|T| \leqq n$. Then $|N(T)| \leqq n - 1$. Without loss of generality, let $n \in N \backslash N[T]$ and $G' := G - \{n\} = (N', A')$. Then $G'$ contains as a subgraph one of the graphs shown in Figs. 7(b), (c), or (d). In particular, $G'$ has two nodes, $k$, $l$, such that $\deg_T^+(k) = \deg_T^-(l) = 0$. We have $T' = T$, $C_1' = C_1$, and by the induction hypothesis (16) defines a facet of $P^*(G')$. So there exists a set of $q := p^2 - 3p + 1$ affinely independent extreme tours in $G'$ (where $p = n - 1$), say $y^i$, $i = 1, \cdots, q$. We construct two sets of affinely independent extreme tours $x^i$ in $G$, each of size $q$, by inserting $n$ after $k$ (for the first set), and before $l$ (for the second set), respectively, into each tour $y^i$. Let $X_1$ and $X_2$ denote the matrices whose rows are the two sets of tours $x^i$. We notice that $2p - 1$ arcs, namely $(k, j), j \in N' \backslash \{k\}$, $(h, n)$, $h \in N' \backslash \{k\}$, and $(n, k)$, have not been used by any of the first $q$ tours. Let $X_{22}$ be the submatrix consisting of the corresponding columns of $X_2$. We notice that the columns $(h, n)$, $h \in N' \backslash \{k\}$, of $X_{22}$ are copies of the columns $(h, l)$, $h \in N' \backslash \{l\}$, of $X_2$ and hence, by Theorem 4.1, the rank of $X_{22}$ is $\geqq 2p - 5$; so we need three additional tours.

We notice that none of the first $2q$ tours has used either of the three arcs $(n, k)$, $(l, n)$, $(k, l)$. A first extreme tour in $G$, containing $(k, l)$ but none of the other two arcs, can be obtained by inserting $n$ into any arc $(i, j) \notin T \cup C_1$, $i \neq l$, $j \neq k$, of any tour $y^i$, $i \in \{1, \cdots, q)$, that contains $(k, l)$. Such a tour is guaranteed to exist by the fact that $|N'| \geqq 6$. A second extreme tour in $G$, containing $(l, n)$ but not $(n, k)$, can be obtained by inserting $n$ into any arc $(l, j) \notin T \cup C_1$, $j \neq k$, of some extreme tour $y^i$. Finally, a third extreme tour in $G$, containing $(n, k)$, can be obtained by inserting $n$ into some arc $(i, k) \notin T \cup C_1$, $i \neq l$, of any extreme tour $y^i$. Again, the existence of tours $y^i$ with the postulated properties follows from the fact that $|N'| \geqq 6$. This completes the proof of Case 5.

The five cases considered are exhaustive. Indeed, the complete digraph $G$ on $n$ nodes for any $n \geqq 6$, with any odd CAT $T$, can be obtained from the digraphs whose node set and odd CATs are shown in Figs. 5, 6(b), 6(c), or 7(a), 7(b), 7(c), by recursive application (in the reverse direction) of the operations used in Cases 1–5 of the induction. For $n = 6$ and $|T| = 7$, we are either in the case shown in Fig. 6(b) or 6(c) (dealt with in Proposition 4.3), or else in the situation shown in Fig. 7(d), which can be reduced by one application of Case 4 of the induction step to the situation in Fig. 7(a) (settled in Proposition 4.2). For $n = 6$ and $|T| = 9$, the situation shown in Fig. 7(e) can be reduced to that of Fig. 6(b) or (c) by one application of Case 1, and so on. To see that the cases considered are exhaustive, note the following.

If $|N[T]| = n \geqq 6$ and we are not in Case 1, 2, or 3, then $T$ either has a single neutral 2-cycle, or else has two that are connected (in $T$) by a path of length at least 3 (this is so because the number of neutral 2-cycles is always odd). In either case, $T$ contains the configuration of Fig. 11, i.e., we are in Case 4. On the other hand, if $|N[T]| \leqq n - 1$ and $n \geqq 7$ we are in Case 5.

This completes the proof of the fact that the inequality (16) defines a facet of $P^*$. Since every tour used in the proof is also a point in $P$, (16) also defines a facet of $P$. □

**5. Relation to earlier work.** Several classes of valid, and sometimes facet-defining, inequalities for the traveling salesman polytope on a directed graph are known. For a thorough survey of the relevant literature see Grötschel and Padberg [1985].

First, if

$$\sum (\alpha_{ij} y_{ij} : (i,j) \in E) \leqq \alpha_0$$

is a valid inequality for the *TS* polytope on an undirected graph, then

$$\sum (\alpha_{ij}(x_{ij} + x_{ji}) : (i,j) \in A, i<j) \leqq \alpha_0$$

is a valid inequality for the ATS polytope on the corresponding directed graph. Thus the various classes of facet-defining inequalities for the TS polytope on an undirected graph, like the subtour elimination inequalities (Dantzig, Fulkerson and Johnson [1954]), the comb inequalities (Chvatal [1973b], Grötschel and Padberg [1979]), the clique tree inequalities (Grötschel and Pulleyblank [1985]) have their correspondents as valid inequalities for the ATS polytope on a directed graph. The subtour elimination inequalities are facet defining for $P^*$ for $n \geqq 5$ and all subtours of length $l$, $2 \leqq l \leqq n - 2$; comb inequalities with exactly three teeth, each containing a single node of the handle, are known to be facet-defining for $\tilde{P}^*$ for $n \geqq 6$, but whether the more general comb inequalities are facet-defining for $\tilde{P}^*$, and whether any comb inequalities are facet-defining for $P^*$, was until recently an open question (except for $n = 6$, in which case they are not). Very recently Fischetti [1988] showed that all comb inequalities are facet-defining for $\tilde{P}^*$ for $n \geqq 6$ and for $P^*$ for $n \geqq 7$. As to the more general clique tree inequalities, it is not known at this point whether they are facet-defining for either $\tilde{P}^*$ or $P^*$.

All the above classes share the feature that the inequalities belonging to them are *symmetric* in the sense that an arc $(i, j)$ belongs to the support of such an inequality if and only if $(j, i)$ does. The inequalities associated with odd CATs do not have this property except for some special cases, so they are distinct from each of the above classes. As to those special cases, they arise when a subset of $T$ together with the chords of type 1 form a complete digraph, in which case this complete digraph becomes the handle of a comb whose teeth are the directed 2-cycles of $T$. Such is the case, for instance, with the odd CAT of length 9 shown in Fig. 7(f). More generally, all comb inequalities corresponding to combs whose handle $H$ and teeth $T_i$, $i = 1, \cdots, k$, satisfy $|T_i| = 2$ for all $i$ and $H \subset (T_1 \cup \cdots \cup T_k)$, are special cases of odd CAT inequalities and thus, from Theorem 4.5, for $|H| \geqq 5$ define facets of $P^*$.

Several classes of asymmetric valid inequalities for the *TS* polytope on a digraph have been identified by Grötschel [1977] and Grötschel and Wakabayashi [1981a], [1981b]. Some of these are derived by lifting the (weak) subtour elimination inequalities. As the support of each such inequality contains a subtour, the odd CAT inequalities, whose support contains no subtour except for some special cases, are obviously distinct from this class. Other classes are associated with hypohamiltonian and hyposemihamiltonian graphs; again, these do not subsume the odd CAT inequalities. Finally, the classes known as $T_k$-inequalities and $C_2$-inequalities (Grötschel [1977]) overlap with the

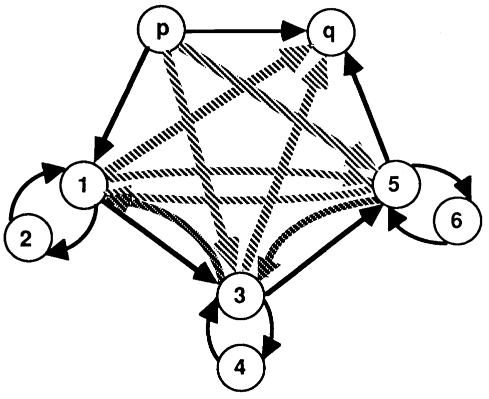


FIG. 12

odd CAT inequalities in that the $T_k$-inequality for $k = 2$ is precisely the odd CAT inequality on four nodes depicted in Fig. 5, and the $C_2$-inequalities with handle $H$ and teeth $T_i$, $i = 1, \cdots, k$, such that $|T_i| = 2$ for all $i$ and $H \subset (T_1 \cup \cdots \cup T_k)$, are precisely the odd CAT's with $2k + 2$ nodes and $k$ neutral 2-cycles, together with their chords of type 1 (see Fig. 12 for the case $k = 3$). For this latter class, the question of whether they are facet-defining for $P$ or $\tilde{P}^*$ (settled in the affirmative in this paper) was still open at the time of writing of Grötschel and Padberg [1985].

**6. Conclusion.** We have given a partial linear description of the asymmetric assignment polytope $P$ defined on a digraph $G$ by identifying a family of valid inequalities associated with odd closed alternating trails of $G$. These inequalities are facet-defining for $P$ and also for the traveling salesman polytope $P^*$ on $G$.

It is to be expected that these inequalities will provide improved bounds and enhanced solution procedures for the ATSP when used as cutting planes either in the context of a pivoting algorithm like that of Padberg and Hong [1980], or in the context of a Lagrangian-based algorithm that takes the cuts in the objective function with appropriate multipliers, as that of Balas and Christofides [1981].

**Acknowledgments.** Thanks are due to Matteo Fischetti and Giovanni Rinaldi for helpful comments on earlier versions of this paper.

## REFERENCES

E. BALAS AND N. CHRISTOFIDES, *A restricted Lagrangian approach to the traveling salesman problem*, Math. Programming, 21 (1981), pp. 19–46.

E. BALAS AND E. ZEMEL, *Lifting and Complementing Yields All the Facets of Positive 0-1 Programming Polytopes*, R. W. Cottle, M. L. Kelmanson, and B. Korte, eds., Math. Programming, Elsevier, North-Holland, Amsterdam, 1984, pp. 13–24.

V. CHVATAL, *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Math., 4 (1973), pp. 305–337.

———, *Tough graphs and Hamiltonian circuits*, Discrete Math., 5 (1973), pp. 215–228.

G. B. DANTZIG, R. D. FULKERSON, AND S. M. JOHNSON, *Solution of a large-scale traveling salesman problem*, Oper. Res., 2 (1954), pp. 393–410.

M. FISCHETTI, *Facets of the asymmetric traveling salesman polytope*, University of Bologna, May 1988.

M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, Freeman, San Francisco, 1979.

M. GRÖTSCHEL, *Polyhedrische Charakterisierungen kombinatorischer Optimierungsprobleme*, Hain, Meisenheim am Glan, 1977.

M. Grötschel and M. Padberg, *On the symmetric traveling salesman problem I-II*, Math. Programming, 16 (1979), pp. 265–280; pp. 281–302.

———, *Polyhedral theory*, E. Lawler, K. Lenstra, A. H. G. Rinnooy Kan and D. Shmoys, eds., The Traveling Salesman Problem, John Wiley, New York, 1985, pp. 251–305.

M. Grötschel and W. Pulleyblank, *Clique tree inequalities and the symmetric traveling salesman problem*, Math. Oper. Res., 11 (1986), pp. 537–569.

M. Grötschel and Y. Wakabayashi, *On the structure of the monotone asymmetric traveling salesman polytope, I: Hypohamiltonian facets*, Discrete Math., 34 (1981), pp. 43–59.

———, *On the structure of the monotone asymmetric traveling salesman polytope. II: Hypotraceable facets*, Math. Programming Stud., 14 (1981), pp. 77–97.

M. Padberg, *On the facial structure of set packing polyhedra*, Math. Programming, 5 (1973), pp. 199–216.

M. Padberg and S. Hong, *On the symmetric traveling salesman problem: A computational study*, Math. Programming Stud., 12 (1980), pp. 78–107.

G. Nemhauser and L. Trotter, *Properties of vertex packing and independence systems polyhedra*, Math. Programming, 6 (1974), pp. 48–61.

S. Sahni, Computationally related problems, SIAM J. Comput., 3 (1974), pp. 262–279.

O. Vornberger, *Komplexität von Wegeproblemen in Graphen*, Ph.D. Thesis, Gesamthochschule Paderborn, West Germany, 1980.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 452–455, November 1989

# INDUCED SUBGRAPHS OF THE POWER OF A CYCLE*

JEAN-CLAUDE BERMOND† AND CLAUDINE PEYRAT†

**Abstract.** In this article, it is shown that if $G$ is an induced subgraph of the $d$th power of a cycle of length $n$, and $G$ has minimum degree $d + k$, then $G$ has at least $[(d + k)/2d]n$ vertices. This answers a problem of Kézdy.

**Key words.** graphs, powers, cycles, degree

**AMS(MOS) subject classification.** 05C99

Let $C_n^d$ be the $d$th power of a cycle on $n$ vertices, that is, the graph whose vertices are the integers modulo $n$, two vertices being joined if and only if their difference is less than or equal to $d$.

During the problem session at the SIAM Conference in San Francisco in June 1988, Kézdy [5] proposed the following problem: Find the minimum number of vertices of any induced subgraph $G$ of $C_n^d$, of minimum degree $d + 1$. In particular, he asked to prove that such a graph must have at least $n/2$ vertices.

This question is motivated by the study of the $d$-girth of a graph. The $d$-girth of a graph $G$ is defined, according to [6], as the minimum number of vertices of an induced subgraph of $G$ with minimum degree $d$ (note that the 2-girth is nothing but the girth). Erdös originally proposed the problem of determining the largest $d$-girth of a graph on $n$ vertices and $m$ edges. Kézdy and Markert [6] considered this problem, and in order to find graphs with a large $(d + 1)$-girth, asked to determine the $(d + 1)$-girth of $C_n^d$. In [6], they gave constructions that show that the $(d + 1)$-girth of $C_n^d$ is at most $(n/2)[1 + (d + 1)/(d^2 + d - \varepsilon)]$, where $\varepsilon = 0$ or 1 whenever $d + 1 \equiv 0$ or $1 \pmod 2$, respectively. They conjectured that this value is exactly the $(d + 1)$-girth of $C_n^d$ and furthermore showed that the $(d + 1)$-girth of $C_n^d$ is at least $n(\sqrt{2} - 1)$.

In this article we show that the $(d + 1)$-girth of $C_n^d$ is at least $[(d + 1)/(2d)]n$, solving the problem proposed in [5] and proving the above conjecture when $d + 1$ is even. In fact, we solve a slightly more general problem by determining the minimum order of an induced subgraph of $C_n^d$ of degree $d + k$ (with $k \geqq 1$).

Note that a minimum induced subgraph of minimum degree $d - k$ (with $k \geqq 0$) of $C_n^d$ is a complete graph on $d - k + 1$ vertices.

Note also that these graphs $C_n^d$, which are particular circulant graphs, have nice properties. For example, they are the graphs with $n$ vertices and connectivity $2d$ with the smallest possible number of edges (see Berge [1]). They also have "super line-connectivity properties" (see Boesch and Wang [2]). They are also important in the design of fault-tolerant networks. For example Chartrand and Kapoor [3], Hayes [4] and Wong and Wong [7] proved that they are minimum $2d - 2$ Hamiltonian graphs (respectively, $2d - 2$ edge-Hamiltonian). That is, they are the graphs with the minimum number of edges among the graphs on $n$ vertices that remain Hamiltonian after the deletion of up to $2d - 2$ vertices (respectively, up to $2d - 2$ edges).

* Received by the editors August 5, 1988; accepted for publication (in revised form) March 31, 1989.

† Informatique, Centre National de la Recherche Scientifique, 3 rue A. Einstein, 06560 Valbonne, France. This note was written while the authors were visiting the School of Computer Science, Simon Fraser University, Burnaby, British Columbia, V5A 1S6, Canada. Support from Advanced Systems Foundations and Simon Fraser University is gratefully acknowledged.

THEOREM. *Let $G$ be an induced subgraph of $C_n^d$ of minimum degree $d+k$, $1 \leqq k \leqq d$, then*

$$\frac{|V(G)|}{n} \geqq \frac{d+k}{2d}.$$

*Proof.* Denote by $V$ the set of the vertices of $G$ and by $\bar{V}$ the complement of this set in the interval $[0, n-1]$.

Let $I = [u, v]$ be an interval of cardinality less than or equal to $d$, with $v$ in $V$. First observe that if $I$ contains $\bar{n}$ elements of $\bar{V}$, then the interval $[v+1, v+d+1]$ must contain at least $\bar{n}+k$ elements of $V$ to insure that the degree of $v$ in $G$ is at least $d+k$.

Now we can construct an infinite sequence of consecutive intervals of integers modulo $n$: $J_1 = [a_1+1, a_2]$, $J_2 = [a_2+1, a_3], \cdots, J_j = [a_j+1, a_{j+1}], \cdots$, satisfying the following properties:

$a_j$ belongs to $V$, for all $j, j \geqq 2$;

$|J_j| \leqq d$, for all $j, j \geqq 1$;

$n_{j+1} = \bar{n}_j + k$, for all $j, j \geqq 1$, where $n_j = |J_j \cap V|$ and $\bar{n}_j = |J_j \cap \bar{V}|$.

Indeed, we can define these intervals recursively as follows:

$J_1 = [a_1+1, a_2]$ is any interval of length at most $d$ such that $a_2$ is in $V$.

Suppose we have constructed the sequence up to $J_j = [a_j+1, a_{j+1}]$, with $a_{j+1}$ in $V$. Then, $J_{j+1} = [a_{j+1}+1, a_{j+2}]$, where $a_{j+2}$ is the $\bar{n}_j + k$th clockwise neighbor of $a_{j+1}$. Note that, by the above observation, $a_{j+2}$ is well defined as $a_{j+1}$ has at least $\bar{n}_j$ counterclockwise neighbors in $\bar{N}$. Note also that $J_{j+1}$ has length at most $d$.

As there exist only a finite number of intervals of length at most $d$, there exist two positive integers $m$ and $p$ such that $J_m = J_{m+p}$.

Now, let $I_1 = J_m$, $I_2 = J_{m+1}, \cdots, I_p = J_{m+p-1}$. This sequence of intervals clearly satisfies the following properties:

(a) for all $i$, $1 \leqq i \leqq p$, $|I_i| \leqq d$.

(b) if $n_i = |I_i \cap V|$ and $\bar{n}_i = |I_i \cap \bar{V}|$, then for all $i$ such that $2 \leqq i \leqq p$, $n_i = \bar{n}_{i-1} + k$ and $n_1 = \bar{n}_p + k$.

(c) all the integers modulo $n$ appear in the same number of intervals.

From (b) and (c), we deduce that

$$\frac{|V|}{n} = \frac{\sum_{i=1}^p n_i}{\sum_{i=1}^p (n_i + \bar{n}_i)}.$$

Let

$$N = \sum_{i=1}^p n_i, \text{then} \sum_{i=1}^p \bar{n}_i = N - kp.$$

Therefore

$$\frac{|V|}{n} = \frac{N}{2N-kp} = \frac{1}{2-\frac{kp}{N}}.$$

However, for all $i$, $1 \leqq i \leqq p-1$, $n_i + n_{i+1} = n_i + \bar{n}_i + k \leqq d+k$ by (a), and similarly $n_p + n_1 \leqq d+k$. Therefore

$$N \leqq \frac{p(d+k)}{2} \text{and} \frac{|V|}{n} = \frac{1}{2-\frac{kp}{N}} \geqq \frac{1}{2-\frac{2k}{d+k}} = \frac{d+k}{2d}.$$

*Remark* 1. If $d + k$ is even, then the bound is sharp. Indeed the subgraph obtained by alternatively taking $(d + k)/2$ vertices in $V$ and $(d - k)/2$ vertices in $\bar{V}$ has minimum degree $d + k$. In fact this subgraph is regular of degree $d + k$, and is isomorphic to a power of a cycle.

*Remark* 2. If $d + k$ is odd, then the bound is not sharp.

For instance, if $d = 2$ and $k = 1$, then the smallest subgraph of minimum degree 3 of $C_n^2$ has $\frac{4}{5}n$ vertices. The extremal graph (already given in [6]) is obtained, when 5 divides $n$, by alternatively taking four vertices in $V$ and one in $\bar{V}$. To prove that this subgraph is the smallest, let us show that any vertex in $\bar{V}$ is followed by at least four vertices in $V$. First recall that if a vertex is in $V$, then at least three of its four neighbors are also in $V$. Then note that there cannot be two or more consecutive vertices in $\bar{V}$, as the first vertex in $V$ after the sequence of vertices in $\bar{V}$ would have at most two neighbors in $V$. Let $\bar{v}$ be a vertex in $\bar{V}$. By the preceding remark, $v + 1$ is in $V$ and its two neighbors $v + 2$ and $v + 3$ are also in $V$. Now, as $v + 2$ has already $\bar{v}$ as neighbor in $\bar{V}$, $v + 4$ is also in $V$.

More generally, one can prove that if $k = d - 1$, then

$$(|V(G)|)/n \geqq 2d/(2d+1).$$

An extremal example is obtained by alternatively taking $2d$ vertices in $V$ and one in $\bar{V}$.

If $d = 4$ and $k = 1$, we can prove that the smallest subgraph of $C_n^4$ of minimum degree 5 has $\frac{12}{19}n$ vertices. The extremal example, when 19 divides $n$, corresponds to repetitions of the sequence

$$\bar{v}\ v\ v\ \bar{v}\ v\ v\ v\ \bar{v}\ v\ \bar{v}\ v\ \bar{v}\ v\ v\ v\ \bar{v}\ v\ v\ \bar{v},$$

where $v$ denotes a vertex in $V$ and $\bar{v}$ a vertex in $\bar{V}$. This example is already given in [6], where it is proved to be unique.

If $d = 5$ and $k = 2$, we can prove that the smallest subgraph of $C_n^5$, with minimum degree 7, has $\frac{5}{7}n$ vertices. An extremal graph corresponds to repetitions of the sequence

$$\bar{v}\ v\ v\ v\ \bar{v}\ v\ v\ v\ v\ \bar{v}\ v\ v\ v\ \bar{v}.$$

If $d = 6$ and $k = 1$, Kézdy and Markert proved that the smallest subgraph of $C_n^6$, with minimum degree 7, has $\frac{24}{41}$ vertices. They also showed that there are two extremal sequences:

$$\bar{v}\ \bar{v}\ \bar{v}\ v\ v\ v\ \bar{v}\ \bar{v}\ v\ v\ v\ v\ \bar{v}\ \bar{v}\ v\ \bar{v}\ v\ v\ \bar{v}\ \bar{v}\ v\ v\ v\ v\ \bar{v}\ \bar{v}\ v\ v\ \bar{v}\ v\ \bar{v}\ \bar{v}\ v\ v\ v\ v\ \bar{v}\ \bar{v}\ v\ v\ v$$

and

$$\bar{v}\ \bar{v}\ v\ \bar{v}\ v\ v\ \bar{v}\ v\ v\ \bar{v}\ v\ v\ \bar{v}\ v\ \bar{v}\ \bar{v}\ v\ v\ \bar{v}\ v\ \bar{v}\ v\ v\ v\ \bar{v}\ v\ \bar{v}\ v\ \bar{v}\ v\ \bar{v}\ v\ \bar{v}\ v\ v\ v\ \bar{v}\ v\ \bar{v}\ v\ v.$$

In view of these results, we offer the following conjecture which contains the conjecture of Kézdy and Markert [6], when $k = 1$. They verified it for $d = 2$, 4, 6, and 8 and $k = 1$.

*Conjecture*. The minimum number of vertices of an induced subgraph of $C_n^d$, with odd minimum degree $d + k$, is

$$\frac{nd(d+3-k)}{2(d^2-(k-2)d-k)}.$$

**Acknowledgment.** We thank P. Hell and A. Kézdy for stimulating discussions.

## REFERENCES

[1] C. BERGE, *Graphs and Hypergraphs*, North Holland, Amsterdam, 1973.

[2] F. T. BOESCH AND J. F. WANG, *Super line-connectivity properties of circulant graphs*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 89–98.

[3] G. CHARTRAND AND S. F. KAPOOR, *On hamiltonian properties of powers of special hamiltonian graphs*, Colloq. Math, 29 (1974), pp. 113–117.

[4] J. P. HAYES, *A graph model for fault-tolerance computing systems*, IEEE Trans. Comput., C-25 (1976), pp. 875–884.

[5] A. KÉZDY, *Problem at the problem session of the SIAM meeting in San Francisco*, P. Winkler, ed., June 1988.

[6] A. KÉZDY AND M. MARKERT, *Fragile graphs*, manuscript, 1987.

[7] W. W. WONG AND C. K. WONG, *Minimum k-hamiltonian graphs*, J. Graph Theory, 8 (1984), pp. 155–165.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 456–466, November 1989

# A MATTER OF DEGREE*

G. CHARTRAND†, H. HEVIA‡, O. OELLERMANN†, F. SABA§, AND A. SCHWENK¶

**Abstract.** The concepts of $n$th degrees and $n$th-order odd vertices in graphs are introduced. The first degree of a vertex $v$ in a graph $G$ is the degree of $v$, while the $n$th degree ($n \geqq 2$) of $v$ is the sum of the $(n-1)$st degrees of the vertices adjacent to $v$ in $G$. By a first-order odd vertex in a graph $G$ is meant an (ordinary) odd vertex in $G$, while for $n \geqq 2$, an $n$th-order odd vertex of $G$ is a vertex adjacent to an odd number of $(n-1)$st-order odd vertices. The number of $n$th-order odd vertices, $n = 1, 2, \cdots$, is investigated. A sequence $s_1, s_2, \cdots, s_n, \cdots$ of integers is defined to be a generalized odd vertex sequence if there exists a graph $G$ containing exactly $s_n$ $n$th-order odd vertices for every positive integer $n$. Generalized odd vertex sequences are characterized. Relationships between the $n$th degrees of the vertices of a graph $G$ and the walks of length $n$ in $G$ are described. The analogous problem for digraphs is also discussed.

**Key words.** $n$th-order odd vertex, $n$th degree, generalized odd vertex sequence

**AMS(MOS) subject classification.** 05C99

## 1. *N*th-order odd vertices.

Probably the most basic result in all of graph theory is that the sum of the degrees of the vertices of a graph $G$ is twice the number of edges in $G$. An *odd vertex* of $G$ is one having odd degree. An immediate consequence of the above result is that there is an even number of odd vertices in every graph. (Terms not defined here may be found in [2].)

The object of this article is to introduce generalizations of the degree of a vertex and of an odd vertex and to establish some results concerning these concepts.

By the first *degree* $\deg_1 v$ of a vertex $v$ of a graph $G$ is meant simply the degree of $v$, while for $n \geqq 2$, the $n$th *degree* $\deg_n v$ of $v$ is the sum of the $(n-1)$st degrees of the vertices of $G$ adjacent to $v$. A graph $G$ together with the $n$th degrees, $n = 1, 2, 3$, of its vertices are shown in Fig. 1.

A vertex $v$ of a graph $G$ is called an $n$th-*order odd vertex* of $G$ if $\deg_n v$ is odd. Equivalently, for $n \geqq 2$, a vertex $v$ is an $n$th-order odd vertex if it is adjacent to an odd number of $(n-1)$st-order odd vertices.

The graph $G$ of Fig. 1 has two first-order odd vertices, two second-order odd vertices and no third-order odd vertices. Consequently, this graph has no $n$th order odd vertices for $n \geqq 3$.

We denote the neighborhood of a vertex $v$ in a graph $G$ by $N(v)$. Then, for $n \geqq 2$,

$$(1) \qquad \deg_n v = \sum_{u \in N(v)} \deg_{n-1} u.$$

Thus, in computing $\sum_{v \in V(G)} \deg_n v$, we note that $\deg_{n-1} u$ occurs $\deg u$ ($= \deg_1 u$) times for each $u \in V(G)$, so that for $n \geqq 2$,

$$(2) \qquad \sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} \deg_1 v \deg_{n-1} v.$$

* Received by the editors June 15, 1988; accepted for publication (in revised form) February 16, 1989.

† Department of Mathematics and Statistics, Western Michigan University, Kalamazoo, Michigan 49008-5152. This research was supported in part by Office of Naval Research contract N00014-88-K-0018.

‡ Instituto de Matematicas, Universidad Catolica de Valparaiso, Valparaiso, Chile. This research was supported by the United Nations through the program PNUD-UNESCO.

§ Department of Mathematics and Statistics, Western Michigan University, Kalamazoo, Michigan 49008-5152.

¶ Department of Mathematics and Statistics, Western Michigan University, Kalamazoo, Michigan 49008-5152. This research was supported in part by Office of Naval Research contract N00014-86-K-0566.

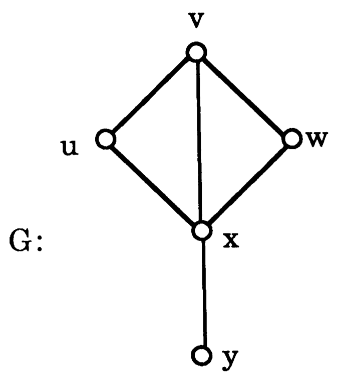


FIG. 1. *The nth degrees of vertices.*

For the purpose of presenting a result involving the number of $n$th-order odd vertices, we consider a generalization of (2).

LEMMA 1. *Let $G$ be a graph and $n \geqq 2$ an integer. Then for every integer $k$ with $1 \leqq k \leqq \lfloor n/2 \rfloor$,*

$$\sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} \deg_k v \deg_{n-k} v. \tag{3}$$

*Proof.* First, we note that (2) shows that the desired result follows for $n = 2$ and $n = 3$. Thus, let $n \geqq 4$. We now proceed by induction on $k$. Again by (2), the lemma holds for $k = 1$.

Let $k$ be an integer with $1 \leqq k < \lfloor n/2 \rfloor$ and assume that

$$\sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} \deg_k v \deg_{n-k} v.$$

Now,

$$\begin{aligned}\sum_{v \in V(G)} \deg_k v \deg_{n-k} v &= \sum_{v \in V(G)} \left(\deg_k v \left(\sum_{u \in N(v)} \deg_{n-k-1} u\right)\right)\\ &= \sum_{u \in V(G)} \left(\deg_{n-k-1} u \left(\sum_{v \in N(u)} \deg_k v\right)\right)\\ &= \sum_{u \in V(G)} \deg_{n-k-1} u \deg_{k+1} u.\end{aligned}$$

Therefore,

$$\sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} \deg_{k+1} v \deg_{n-k-1} v.$$

We now present an extension of the well-known result on the number of odd vertices in a graph, which we mentioned earlier.

THEOREM 1. *For each positive integer $n$, every graph contains an even number of $n$th-order odd vertices.*

*Proof.* The proof is by induction on $n$, noting again that the result is true for $n = 1$. Let $n > 1$ be an integer and assume for every integer $k$ with $1 \leqq k < n$, that $G$ contains an even number of $k$th-order odd vertices. We consider two cases.

*Case* 1. Suppose that $n$ is even, say $n = 2m$. By Lemma 1,

$$\sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} (\deg_m v)^2. \tag{4}$$

By the inductive hypothesis, the right-hand expression in (4) contains an even number of odd terms and so is even. Consequently, the left-hand expression in (4) contains an even number of odd terms, so that $G$ contains an even number of $n$th-order odd vertices.

*Case* 2. Suppose that $n$ is odd, say $n = 2m + 1$. Applying Lemma 1 again, we have

$$\sum_{v \in V(G)} \deg_n v = \sum_{v \in V(G)} \deg_m v \cdot \deg_{m+1} v. \tag{5}$$

Note that a term $\deg_m v \cdot \deg_{m+1} v$ in the right-hand sum in (5) is odd if and only if both $\deg_m v$ and $\deg_{m+1} v$ are odd. Further, $\deg_{m+1} v$ is odd if and only if $v$ is adjacent to an odd number of $m$th-order odd vertices.

If $G$ contains no $m$th-order odd vertices, then $G$ contains no $r$th-order odd vertices for $r > m$ so, in particular, $G$ contains no $n$th-order odd vertices and the desired result follows. Suppose then that $G$ contains (an even number of) $m$th-order odd vertices. Let $H$ be the subgraph induced by the $m$th-order odd vertices. Then both $\deg_m v$ and $\deg_{m+1} v$ are odd if and only if $v$ has odd degree in $H$. Since $H$ has an even number of odd vertices, $G$ contains an even number of vertices $v$ for which $\deg_m v \cdot \deg_{m+1} v$ is odd. Hence the right-hand sum in (5) contains an even number of odd terms, implying that $G$ contains an even number of $n$th-order odd vertices.

**2. Generalized odd vertex sequences.** By Theorem 1, we know that every graph contains an even number of $n$th-order odd vertices for every positive integer $n$. Therefore, for each graph $G$, there exists an infinite sequence $s_1, s_2, \cdots, s_n, \cdots$ of nonnegative even integers where $s_n$ is the number of $n$th-order odd vertices of $G$. In this section, we consider the following question: For which sequences $s_1, s_2, \cdots, s_n, \cdots$ of nonnegative even integers does there exist a graph $G$ such that $G$ contains $s_n$ $n$th-order odd vertices for each positive integer $n$? We shall call such a sequence that is realized by a graph a *generalized odd vertex sequence*.

We begin by establishing two necessary conditions for a sequence of nonnegative even integers to be a generalized odd vertex sequence. Suppose that $\{s_n\}$ is a generalized odd vertex sequence for which $s_k = 0$ for some positive integer $k$. Then this sequence is realized by a graph $G$ that contains no $k$th-order odd vertices. Certainly, then, every vertex of $G$ is adjacent to an even number (namely, zero) of $k$th-order odd vertices, so that $s_{k+1} = 0$. Consequently, if $s_k = 0$, then $s_i = 0$ for all $i > k$.

A sequence $\{s_n\}$ is called *ultimately periodic*, or more simply *periodic*, if there exist positive integers $l$ and $N$ such that $s_k = s_{k+l}$ for all $k \geqq N$.

THEOREM 2. *Every generalized odd vertex sequence is periodic.*

*Proof.* Let $G$ be a graph with $V(G) = \{v_1, v_2, \cdots, v_p\}$ and let $\{s_n\}$ be the generalized odd vertex sequence of $G$. For each positive integer $n$, let $\mathbf{a}_n$ be the $p$-vector $(a_{n,1}, a_{n,2}, \cdots, a_{n,p})$, where $a_{n,i} = 1$ if $v_i$ is an $n$th-order odd vertex and $a_{n,i} = 0$ otherwise. Thus $a_{n+1,i} = (\sum a_{n,j}) (\text{mod } 2)$, where the sum is taken over all $j$ such that $v_j \in N(v_i)$. Since $a_{n,i} = 0$ or 1 for each positive integer $n$ and each $i$ with $1 \leqq i \leqq p$, the number of distinct vectors $\mathbf{a}_n$ is at most $2^p$. Hence there exist positive integers $N$ and $l$ such that $\mathbf{a}_N = \mathbf{a}_{N+l}$, which implies that $s_N = s_{N+l}$. From our observation about $a_{n+1,i}$, it follows that $\mathbf{a}_{N+1} = \mathbf{a}_{N+l+1}$ and that $s_{N+1} = s_{N+l+1}$. Hence, proceeding inductively, we see that $s_k = s_{k+l}$ for all $k \geqq N$.

Let $\{s_n\}$ be the generalized odd vertex sequence of a graph $G$. Since $\{s_n\}$ is periodic, there are integers $N$ and $l$ for which $s_k = s_{k+l}$ for all $k \geqq N$. Let $m$ be the smallest such

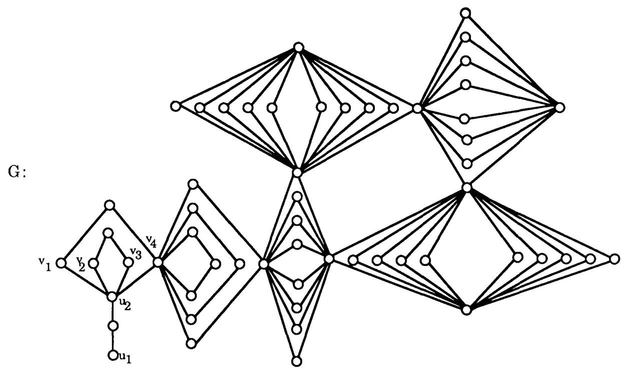


FIG. 2. *A graph with generalized odd vertex sequence* 2, 4, 6, 8, 10, 8, 10, $\cdots$.

$N$ that satisfies this condition. The (finite) sequence $s_m, s_{m+1}, \cdots, s_{m+l-1}$ is called a *period* of $G$ and $l$ the *length of a period*. If $m \geqq 2$, then we refer to $s_1, s_2, \cdots, s_{m-1}$ as the *pre-period* of $G$. While the pre-period, if it exists, of $G$ is unique, a period of $G$ is never unique. For example, if $s$: 2, 4, 6, 8, 10, 8, 10, $\cdots$ is the generalized odd vertex sequence of a graph $G$, then 2, 4, 6 is the pre-period of $G$, while 8, 10 is a period of length 2 of $G$. For every positive even integer $n$, there is a period of length $n$. For instance, 8, 10, 8, 10 is a period of length 4, while 8, 10, 8, 10, 8, 10 is a period of length 6. Figure 2 shows a graph $G$ having $s$ as its generalized odd vertex sequence. The vertices $u_1$ and $u_2$ are the only odd vertices, while $v_1, v_2, v_3$ and $v_4$ are the only second-order odd vertices.

The construction given in Fig. 2 illustrates a general construction used to prove the next result. In order to present a proof, it is convenient to introduce some notation. For a positive integer $n$, the graph $S_n$ denotes the star of size $n$ in which each edge has been subdivided, $D_n$ denotes the complete bipartite graph $K_{2,n}$, and $F_{2n}$ is that graph obtained by the identification of $n$ vertices, one vertex in each of $n$ pairwise disjoint 4-cycles. The graphs $S_4$, $D_4$, and $F_8$ are shown in Fig. 3.

THEOREM 3. *A sequence* $\{s_n\}$ *of nonnegative even integers is the generalized odd vertex sequence of a graph if and only if*:

(i) $\{s_n\}$ *is periodic*, *and*

(ii) $s_k = 0$ *implies* $s_{k+1} = 0$.

*Proof.* If $\{s_n\}$ is a generalized odd vertex sequence of a graph, then $\{s_n\}$ is periodic by Theorem 2. We have further noted that if $s_k = 0$ for some positive integer $k$, then $s_{k+1=0}$.

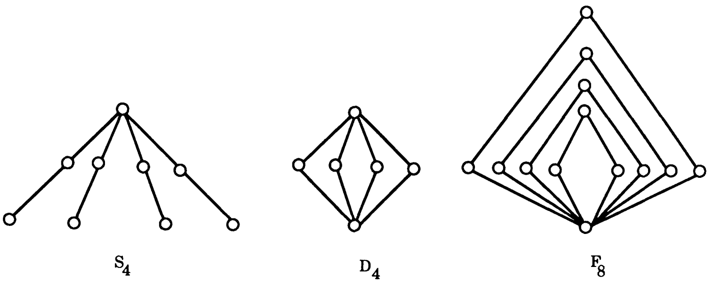


FIG. 3

Conversely, suppose that $\{s_n\}$ is a periodic sequence of nonnegative even integers with the property that if $s_k = 0$ for some positive integer $k$, then $s_{k+1} = 0$. We now consider two cases depending on whether $G$ does or does not have a pre-period.

*Case* 1. Assume that $G$ has a pre-period, say $s_1, s_2, \cdots, s_{m-1}$, where then $m \geqq 2$. Choose a period of $G$ so that its length $l$ is at least 3. Thus $s_m, s_{m+1}, \cdots, s_{m+l-1}$ is a period of $G$. Let $s'_i = s_{m-1+i}$ for $1 \leqq i \leqq l$.

Suppose first that $m = 2$. Define $G_1 \cong S_{s_1-1}$. If $s_2 = 0$, then $G \cong G_1$ has $\{s_n\}$ as its generalized odd vertex sequence. We assume therefore that $s_2 \neq 0$ so that $\{s_n\}$ consists only of positive even integers. Let $G'_1 \cong D_{s'_1}$ and $G'_i \cong F_{s'_i}$ for $2 \leqq i \leqq l$. Further, let $v_1$ be a vertex of degree $s_1 - 1$ in $G_1$, and let $u'_1$ and $u''_1$ be the two vertices of degree $s'_1$ in $G'_1$. (If $s'_1 = 2$, choose $u'_1$ and $u''_1$ to be nonadjacent vertices of degree 2 in $G'_1$.) Let $u$ be some vertex of $G'_1$ different from $u'_1$ and $u''_1$. For $2 \leqq i \leqq l$, let $u'_i$ be a vertex of degree $s'_i$ in $G'_i$ and let $u''_i$ be a vertex of degree 2 in $G'_i$ (that is adjacent to $u'_i$). The graph $G$ is obtained by identifying $v_1$ and $u''_1$, identifying $u$ and $u'_l$, and identifying $u'_i$ and $u''_{i+1}$ for $i = 1, 2, \cdots, l-1$. Then $G$ has $\{s_n\}$ as its generalized odd vertex sequence. (Figure 4 illustrates the construction for the sequence 6, 2, 4, 2, 4, 2, 4, $\cdots$.)

Next we assume that $m \geqq 3$. The graph $G$ then has the pre-period $s_1, s_2, \cdots, s_{m-1}$. Define $G_1 \cong S_{s_1-1}$ and for $2 \leqq i \leqq m-1$, define $G_i \cong F_{s_i}$. Moreover, let $v_1$ be a vertex of degree $s_1 - 1$ in $G_1$, and for $2 \leqq i \leqq m-1$, let $v_i$ be a vertex of degree $s_i$ in $G_i$ and let $v'_i$ be a vertex of degree 2 in $G_i$ (that is adjacent to $v_i$). Define the graph $H$ by identifying $v_1$ and $v_2$ and by identifying $v'_i$ and $v_{i+1}$ for $2 \leqq i \leqq m-2$. If $s_m = 0$, then $H$ has $\{s_n\}$ as its generalized odd vertex sequence. Hence we assume that $s_n \geqq 2$. We now define graphs $G'_1, G'_2, \cdots, G'_l$ as before. We identify $u'_1$ in $G'_1$ with $v'_{m-1}$ in $H$ and identify the vertices of $G'_1, G'_2, \cdots, G'_l$ as described before to produce $G$, which has $\{s_n\}$ as its generalized odd vertex sequence. (Figure 5 illustrates this construction for the sequence 4, 8, 6, 4, 2, 4, 4, 2, 4, $\cdots$.)

*Case* 2. Assume that $G$ has no pre-period. If $s_1 = 0$, then any graph with only even vertices has the desired generalized odd vertex sequence (namely 0, 0, 0, $\cdots$). Hence, we assume that $s_1 \geqq 2$ and consequently, that $s_i \geqq 2$ for all $i \geqq 1$. Again assume that $G$ has period $l \geqq 3$. Define $G_1 \cong K_{1,s_1-1}$ (a star) and $G_i \cong F_{s_i}$ for $2 \leqq i \leqq l$. Let $v_1$ be a vertex of degree $s_1 - 1$ in $G_1$. For $2 \leqq i \leqq l$, let $v_i$ be a vertex of degree $s_i$ in $G_i$ and let $v'_i$ be a vertex of degree 2 in $G_i$ that is adjacent to $v_i$. The graph $G$ is produced by joining $v_1$ and $v_2$, by identifying $v'_i$ and $v_{i+1}$ for $2 \leqq i \leqq l-1$, and by identifying $v'_l$ and $v_1$. Then $G$ has $\{s_n\}$ as its generalized odd vertex sequence. (Figure 6 illustrates this construction for the sequence 6, 8, 2, 4, 6, 8, 2, 4, $\cdots$.)

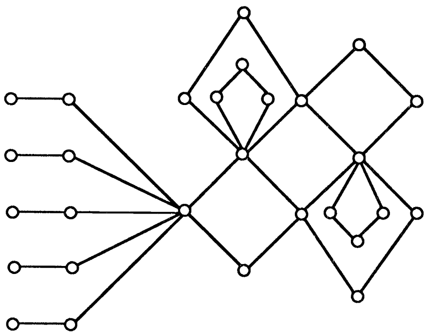


FIG. 4. *A graph with generalized odd vertex sequence* 6, 2, 4, 2, 4, 2, 4, $\cdots$.

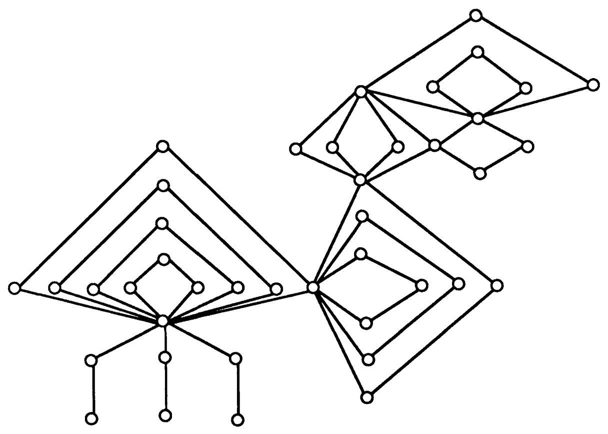


FIG. 5. *A graph with generalized odd vertex sequence* 4, 8, 6, 4, 2, 4, 4, 2, 4, $\cdots$.

According to Theorem 1, then, if a graph $G$ of order $p$ contains $m$ $n$th-order odd vertices (where, of course, $0 \leqq m \leqq p$), then $m$ is even. In connection with this statement, we determine which such triples $m$, $n$, $p$ of integers are realizable.

THEOREM 4. *For all triples $m$, $n$, $p$ of integers with $m$ even, $n$ and $p$ positive, and $0 \leqq m \leqq p$, there exists a connected graph $G$ of order $p$ that contains exactly $m$ $n$th-order odd vertices unless* (1) $p = m + 1 = 3$ *and* $n \geqq 2$ *or* (2) $m = 0$ *and* $p = 2$.

*Proof.* We consider three cases.

*Case* 1. Assume $p = m$. Then $G \cong K_p$ has the desired properties for every positive integer $n$.

*Case* 2. Assume $p = m + 1$. If $m = 0$, then $G \cong K_1$ satisfies the conclusion of the theorem for every integer $n \geqq 1$. If $m \geqq 2$ and $n = 1$, then $G \cong K_{1,m}$ provides the desired result. Hence, it remains to consider $m \geqq 2$ and $n \geqq 2$ in this case.

If $m = 2$, then $p = 3$ and $G \cong K_3$ or $G \cong P_3$. Neither of these graphs contains any $n$th-order odd vertices (for $n \geqq 2$). For $m \geqq 4$ and $n \geqq 2$, the graph $G$ of Fig. 7 has the desired properties.

*Case* 3. Assume $p \geqq m + 2$. First we consider $m = 0$. If $p = 2$, then $G \cong K_2$ is the only connected graph and $G$ has two $n$th-order odd vertices for all integers $n \geqq 1$, so that an exception is produced here. If $p \geqq 3$, then $G \cong C_p$ has the desired properties. Hence

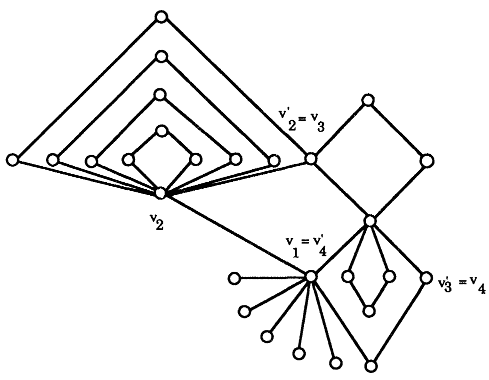


FIG. 6. *A graph with generalized odd vertex sequence* 6, 8, 2, 4, 6, 8, 2, 4, $\cdots$.

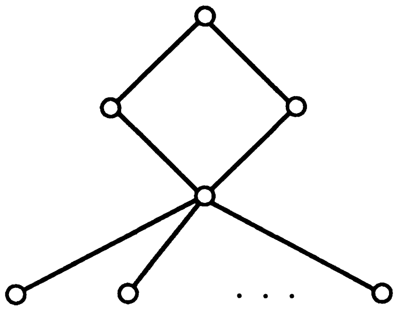


FIG. 7

we may assume that $m \geqq 2$, so that $p \geqq m + 2 \geqq 4$. We consider $p$ odd and $p$ even separately.

*Subcase* 3.1. Assume $p$ is odd. If $m = 2$, the graph $G_1$ of Fig. 8(a) has the appropriate properties, while if $m \geqq 4$, the graph $G_2$ of Fig. 8(b) has the desired properties.

*Subcase* 3.2. Assume $p$ is even. To obtain a graph $G$ with the desired properties here, let $G = G_1 - uv$ if $m = 2$, where $G_1$ is the graph of Fig. 8(a), and let $G = G_2 - vu$ if $m \geqq 4$, where $G_2$ is the graph of Fig. 8(b).

**3. The matrix point of view.** For a graph $G$ with vertex set

$$V(G) = \{v_1, v_2, \cdots, v_p\}$$

and adjacency matrix $A$, the $(i,j)$ entry of $A^n$ is known to be the number of $v_i - v_j$ walks of length $n$ in $G$. The (ordinary) degree of $v_i$ can be viewed as the number of walks of length 1 that begin at $v_i$. Inductively, the $n$th degree of $v_i$ is the number of walks of length $n$ that begin at $v_i$. Thus, $\deg_n v_i$ is the sum of the entries in the $i$th row of $A^n$. The product of $A^n$ and the column vector $\mathbf{u} = (1, 1, \cdots, 1)^T$ produces a vector whose $i$th entry is $\deg_n v_i$. With this perspective, (1) is the entry corresponding to vertex $v$ in the vector identity

$$A^n\mathbf{u} = A(A^{n-1}\mathbf{u}). \tag{6}$$

Similarly, multiplying (6) on the left by $\mathbf{u}^T$ produces (2), so that Lemma 1 becomes the matrix identity

$$\mathbf{u}^T(A^n\mathbf{u}) = (A^k\mathbf{u})^T A^{n-k}\mathbf{u}. \tag{7}$$

Theorem 1 implies that the total number of walks of length $n$ must be even. We now discuss an alternative proof of Theorem 1. We proceed by induction on $n$. Since $\sum_{v \in V(G)} \deg_1 v$ is even, the result follows for $n = 1$. Assume that $n \geqq 2$ and that the result holds for all $k$ $(1 \leqq k < n)$. Pair each walk $w_0, w_1, \cdots, w_n$ with its "reversal" $w_n, w_{n-1}, \cdots, w_0$. Thus, the number of walks, each of which does not equal its reversal,

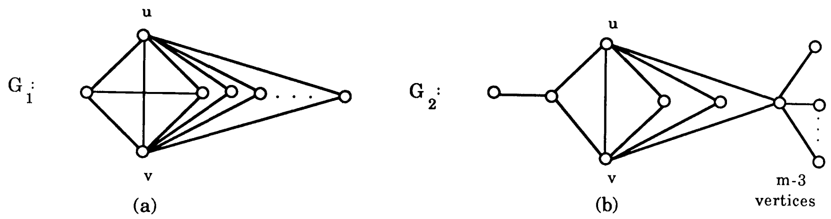


FIG. 8

is even. A walk $W$: $w_0, w_1, \cdots, w_n$ equals its reversal if and only if $w_i = w_{n-i}$ for each $i$ $(0 \leqq i \leqq n)$. When $n$ is odd, say $n = 2k + 1$, the walk $W$ can never equal its reversal, for otherwise $v_k$ both equals and is adjacent to $v_{k+1}$. On the other hand, if $n$ is even, say $n = 2k$, the walk $W$ equals its reversal if and only if $W$ is formed by a walk of length $k$ followed by its reversal. The number of such walks $W$ is given by $\mathbf{u}^T A^k \mathbf{u}$. Since $k = n/2 < n$, we know from the inductive hypothesis that the number of these walks $W$ is also even.

We now characterize graphs whose $n$th degrees are all equal, that is, the *nth degree regular graphs*. Of course, first-degree regular graphs are regular graphs. A graph is *biregular* if it is bipartite with partite sets $V_1$ and $V_2$ such that all vertices in $V_1$ have a common degree $d_1$ and all vertices in $V_2$ have common degree $d_2$. Thus, a biregular graph is regular if $d_1 = d_2$. For example, the complete bipartite graphs $K_{m,n}$ are biregular. Observe that biregular graphs are also second degree regular of degree $d_1 d_2$.

The following statement appears without proof in Plonka [3].

THEOREM 5. *Let n be a positive integer. If a connected graph G is nth degree regular, then either*

(i) *G is regular, or*

(ii) *n is even and G is biregular.*

*Proof.* If $G$ is $n$th degree regular, say of degree $t$, then

$$A^n \mathbf{u} = t\mathbf{u}.$$

Now if $G$ is regular, say of degree $r$, then $A\mathbf{u} = r\mathbf{u}$, and so repeated multiplication shows that $t$ must equal $r^n$. But if $G$ is not regular, then $\mathbf{u}$ cannot be an eigenvector of $A$ (see Schwenk and Wilson [4; p. 313, Cor. 4.4]). Nevertheless, there exists an orthonormal basis $\mathbf{X}_1, \mathbf{X}_2, \cdots, X_{\mathbf{p}}$ of eigenvectors (with eigenvalues $\lambda_1 \geqq \lambda_2 \geqq \cdots \geqq \lambda_p$ and $\lambda_1 \geqq |\lambda_i|$ for all $i$). Write $\mathbf{u} = \sum c_i \mathbf{X}_i$ and substitute into (7) to obtain $\sum c_i \lambda_i^n \mathbf{X}_\mathbf{i} = \sum t c_i \mathbf{X}_\mathbf{i}$. For each $i$, we multiply (dot product) the equation with $\mathbf{X}_\mathbf{i}$ to obtain $c_i \lambda_i^n = t c_i$. In other words, $c_i(\lambda_i^n - t) = 0$ for every $i$. The Perron-Frobenius Theorem [1, Thm. 4.1, p. 312] guarantees that in a connected graph, each entry of the first eigenvector is strictly positive and $\lambda_1$ has multiplicity 1. Therefore, $c_1 = \mathbf{u} \cdot \mathbf{X}_1 > 0$ and so $t = \lambda_1^n$. If $c_i = 0$ for all $i \geqq 2$, then $\mathbf{u} = c_1 \mathbf{X}_1$ is an eigenvector and $G$ is regular. On the other hand, if $c_i \neq 0$ for some $i \geqq 2$, then $\lambda_1^n = \lambda_i^n$. Now $\lambda_1$ is simple, so $n$ is even and $\lambda_i = -\lambda_1$. But the "pairing theorem" [4, Thm. 4.2, p. 312] assures that $-\lambda_1$ is an eigenvalue of $A$ if and only if $G$ is bipartite. Furthermore, in this case, $\lambda_i = -\lambda_{p+1-i}$. Thus, the simplicity of $\lambda_1$ implies that $\lambda_p = -\lambda_1$ is also simple. We have

$$\begin{aligned} A^n \mathbf{u} &= A^n(c_1 \mathbf{X}_1 + c_p \mathbf{X}_\mathbf{p}) \\ &= c_1(\lambda_1)^n \mathbf{X}_1 + c_p(-\lambda_1)^n \mathbf{X}_\mathbf{p} \\ &= \lambda_1^n (c_1 \mathbf{X}_1 + c_p \mathbf{X}_\mathbf{p}) \\ &= \lambda_1^n \mathbf{u}. \end{aligned}$$

Next, suppose that $\mathbf{X}_1 = (a_1, a_2, \cdots, a_p)$, where then each $a_i > 0$. The pairing theorem also guarantees that in a bipartite graph, $\mathbf{X}_\mathbf{p}$ is obtained from $\mathbf{X}_1$ by negating the entries in exactly one of the partite sets. Thus,

$$\mathbf{X}_\mathbf{p} = (a_1, a_2, \cdots, a_m, -a_{m+1}, \cdots, -a_p).$$

Now consider the unit vector $e_i = (0, 0, \cdots, 0, 1, 0, 0, \cdots, 0)$ which has its only 1 in the $i$th position. For every $i \leqq m$, then $\mathbf{e}_\mathbf{i} \cdot \mathbf{u} = \mathbf{e}_\mathbf{i} \cdot (c_1 \mathbf{X}_1 + c_p \mathbf{X}_\mathbf{p})$, implying that $1 =$

$(c_1 + c_p)a_i$. That is, for all $i \leqq m$, we have $a_i = (1/c_1 + c_p)$. Similarly, for all $i > m$, $a_i = 1/(c_1 - c_p)$. Finally, $A\mathbf{X}_1 = \lambda_1\mathbf{X}_1$ requires that each vertex in the first partite set has degree $d_i$ where $d_i(1/(c_1 - c_p)) = \lambda_1(1/(c_1 + c_p))$. That is, all vertices in partite set $V_1$ have common degree $d_1 = \lambda_1((c_1 - c_p)/(c_1 + c_p))$. Similarly, the entries in the partite set $V_2$ have common degree $d_2 = \lambda_1((c_1 + c_p)/(c_1 - c_p))$. Thus, $n$th degree regularity implies either regularity or biregularity.

**4. Directed graphs.** For directed graphs, we define the $n$th outdegree and the $n$th indegree of a vertex. The *nth outdegree* of a vertex $v_i$ $(1 \leqq i \leqq p)$ is the number of walks of length $n$ that begin at $v_i$, while the *nth indegree* of $v_i$ is the number of walks of length $n$ that terminate at $v_i$. While these need not be equal, their properties are analogous. Thus, we restrict our attention to $n$th outdegrees. There is a directed analogue to Theorem 5. In order to present this result, we begin by defining a few new terms.

A digraph is said to be *m-cyclic* if its vertices can be partitioned into $m$ sets $V_1$, $V_2, \cdots, V_m$ so that each vertex in the set $V_i$ $(1 \leqq i \leqq m)$ is adjacent only to vertices in $V_{i+1}$ (where indices are expressed modulo $m$). An $m$-cyclic digraph is called *m-cyclic-outregular* if for each $i$, all vertices in $V_i$ have a common outdegree $d_i$. For $m = 1$, a 1-cyclic-outregular digraph is, in fact, an empty digraph and thus a 1-outregular digraph. Observe that $m$-cyclic-outregularity implies $n$th outregularity for every $n$ divisible by $m$, since for $n = km$ the $n$th outdegree is just $(d_1 d_2 \cdots d_m)^k$ for every vertex of the digraph. Syslo [5] credits the following theorem to McKay in an unpublished work. We find the result sufficiently significant to merit a proof in print. Some additional terms will be useful.

For a given matrix $A$, the *digraph underlying A* has an arc $v_i v_j$, whenever $a_{ij} \neq 0$. The matrix is called *irreducible* if its underlying digraph is strongly connected; otherwise $A$ is *reducible*. The Perron–Frobenius Theorem (see Berman and Plemmons [1, Thm. 1.4, p. 27]) states that the first eigenvector will be strictly positive if $A$ is irreducible and will be nonnegative if $A$ is reducible.

THEOREM 6. *Let D be a strongly connected digraph. If D is nth outregular, then there exists a positive integer divisor m of n such that D is m-cyclic outregular.*

*Proof.* Since $D$ is $n$th outregular, $\mathbf{u} = (1, 1, \cdots, 1)^T$ is an eigenvector of $A^n$. Let $A^n\mathbf{u} = t\mathbf{u}$ $(n > 1)$. If $\mathbf{u}$ happens to be an eigenvector of $A$ as well, then $D$ is outregular and we are done. Otherwise, let $\mathbf{X}_1$ be the (unique) positive eigenvector of $A$ guaranteed by the Perron–Frobenius Theorem. Now $A^n\mathbf{X}_1 = \lambda_1^n\mathbf{X}_1$ so we have found a second positive eigenvector for $A^n$.

Therefore, $A^n$ is reducible (even though the digraph underlying $A$ is strongly connected, and hence $A$ is irreducible). Let $B_o$ be one block of $A^n$. Define a partition of the vertices by setting $B_i$ to be all vertices at distance $i$ from $B_o$. Suppose that this gives sets $B_o, B_1, \cdots, B_{m-1}$. Now each vertex in $B_i$ is adjacent from at least one vertex in $B_{i-1}$. Since $B_o$ is a block of $A^n$, then all walks of length $n$ that start in $B_o$ must also end in $B_o$. This implies that $m$ divides $n$. But even stronger, all arcs in $D$ must join a vertex in $B_i$ to one in $B_{i+1}$, where indices are expressed modulo $m$. If this were not so, let $i$ be the smallest integer such that $B_i$ contains a vertex $v$ that is adjacent to some vertex $w$ in $B_j$ with $j \neq i + 1$. If $j > i + 1$, then $w$ is adjacent from a vertex in $B_i$ so that $w$ should be in $B_{i+1}$, a contradiction. If $j \leqq i$, then $w$ can be reached from $B_o$ in $j$ steps and also in $i + 1$ $(> j)$ steps. Continuing for $m - j$ additional steps, we have a walk of length $m$ from $B_o$ to $B_o$ and another walk of length $m$ from $B_o$ to $B_{i+1-j}$. But then $B_o$ is not a block of $A^n$, contrary to its initial definition. Thus, the digraph $D$ underlying $A$ has the structure shown in Fig. 9. All arcs with initial vertices in $B_i$ have their terminal vertices in $B_{i+1}$.

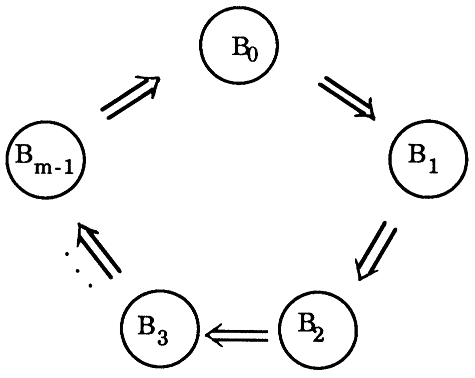


FIG. 9

The adjacency matrix $A$ of the digraph of Fig. 9 has the structure

$$A = \begin{bmatrix} 0 & A_{0,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{1,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{2,3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdot^{\cdot^{\cdot}} \\ A_{m-1,0} & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $A_{i,i+1}$ indicates the arcs that join vertices of $B_i$ to vertices of $B_{i+1}$. Now $A^n$ has its nonzero blocks appearing only on the main diagonal.

Each block $B_i$ of $A^n$ is irreducible since $A$ is strongly connected. Thus each has a unique positive eigenvector. However, $A^n\mathbf{u} = t\mathbf{u}$ and $A^n\mathbf{X}_1 = \lambda_1^n\mathbf{X}_1$. Thus, within blocks, $\mathbf{X}_1$ must be a scalar multiple of $\mathbf{u}$. Equivalently, $\mathbf{X}_1$ is constant for each block,

$$\mathbf{X}_1 = (a_0, \cdots, a_0, a_1, \cdots, a_1, a_2, \cdots, a_2, \cdots, a_{m-1}, \cdots, a_{m-1})^T.$$

Now for each vector $\mathbf{v}$ in a particular block $B_i$, the equation $A\mathbf{X}_1 = \lambda_1\mathbf{X}_1$ implies that

$$odv \cdot a_{i+1} = \lambda_1 a_i.$$

Thus, all vertices in $B_i$ have the common outdegree $\lambda_1 a_i / a_{i+1}$. Since this holds for every block $B_i$, the digraph is $m$-cyclic-outregular, as claimed in the theorem.

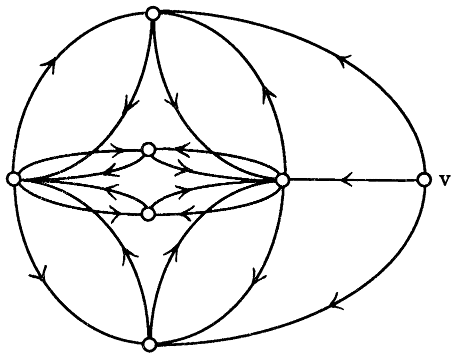


FIG. 10

If the digraph $D$ of Theorem 6 is not strongly connected, it can still be $n$th outregular by having an $m$-cyclic-outregular subdigraph $S$ along with additional vertices whose arcs lead into $S$ with an appropriate number of walks of length $n$. For example, the digraph $D$ shown in Fig. 10 is second outregular with $r = 8$. The subdigraph $S = D - v$ is 2-cyclic-outregular with outdegrees 2 and 4. Thus, starting at any vertex in $S$ there are $2 \cdot 4$ $(=8)$ walks of length 2. But $v$ has been joined to three vertices in $D$ so that $v$ has second outdegree equal to 8. A variation of this construction can be used for each $n$ to form a digraph with $n$ different outdegrees that is, nevertheless, second-degree regular.

## REFERENCES

[1] A. Bermann and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979, pp. 26–32.

[2] G. Chartrand and L. Lesniak, *Graphs & Digraphs*, Second edition, Wadsworth & Brooks/Cole, Monterey, CA, 1986.

[3] J. Plonka, *On Γ-regular graphs*, Colloq. Math., 46 (1982), pp. 131–134.

[4] A. J. Schwenk and R. J. Wilson, *On the eigenvalues of a graph*, Selected Topics in Graph Theory, L. W. Beineke and R. J. Wilson, eds., Academic Press, London, 1978, pp. 307–336.

[5] M. M. Syslo, *Adjacency matrix equations and related problems: Research notes*, Commentationes Mathematicae Universitatis Carolinae 24, 2 (1983), pp. 211–222.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 467–472, November 1989

# PEBBLING IN HYPERCUBES*

FAN R. K. CHUNG†

**Abstract.** This paper considers the following game on a hypercube, first suggested by Lagarias and Saks. Suppose $2^n$ pebbles are distributed onto vertices of an $n$-cube (with $2^n$ vertices). A pebbling step is to remove two pebbles from some vertex and then place one pebble at an adjacent vertex. The question of interest is to determine if it is possible to get one pebble to a specified vertex by repeatedly using the pebbling steps from any starting distribution of $2^n$ pebbles. This question is answered affirmatively by proving several stronger and more general results.

**Key words.** distributive lattice, graphs, retracts

**AMS(MOS) subject classification.** 05C

**1. Introduction.** An $n$-dimensional cube, or $n$-cube for short, consists of $2^n$ vertices labelled by (0, 1)-tuples of length $n$. Two vertices are adjacent if their labels are different in exactly one entry. Because of its highly parallel structure, the $n$-cube possesses many nice properties and is an ideal model for games of various distributive types. In this paper we investigate the following game that was first proposed by Lagarias and Saks [4], [7].

Suppose $2^n$ pebbles are distributed onto vertices of an $n$-cube. A pebbling step consists of removing two pebbles from one vertex and then placing one pebble at an adjacent vertex. We say a pebble can be moved to a vertex if we can apply pebbling steps repeatedly (if necessary) so that in the resulting configuration the vertex has one pebble. The question of interest is to determine if it is always possible to move one pebble to a specified vertex from any starting distribution of $2^n$ pebbles.

In this paper we answer this problem affirmatively. Independently, Guzman also solved the same problem by a different proof.

THEOREM 1. *For any distribution of $2^n$ pebbles to vertices of the n-cube, one pebble can be moved to any specified vertex.*

Theorem 1 turns out to be an immediate consequence of some stronger and more general results that lead to an alternative proof for the following result (due to Lemke and Kleitman [5] through a different method).

For any given integers $a_1, a_2, \cdots, a_d$ there is a nonempty subset

$$X \subseteq \{1, 2, \cdots, d\}$$

such that

$$d \mid \sum_{i \in X} a_i \quad \text{and} \quad \sum_{i \in X} gcd(a_i, d) \leqq d.$$

For any partially ordered set, the set of order ideals (downward closed subsets) ordered by inclusion is a distributive lattice. The Hasse diagram of the lattice can be viewed as a graph where the ideals $u$ and $v$ are adjacent if $u$ contains $v$ and $u$ is exactly one larger than $v$. One of many variations of our result is the following.

Consider a given partially ordered set $S$ in which each element is assigned an integer weight. In the corresponding finite distributive lattice, an admissible move involves two adjacent vertices $u$ and $v$, say $u < v$. That is, to remove $w$ pebbles from $v$ (where $w$ is

* Received by the editors August 8, 1988; accepted for publication (in revised form) March 9, 1989.
† Bellcore, Morristown, New Jersey 07960-1910.

the weight of the element in $v$ but not in $u$) and to place one pebble on $u$. If $\prod_{x \in S} w(x)$ pebbles are assigned to vertices in the distributive lattice, then by repeatedly applying the admissible steps, one pebble can be placed on the empty set in the lattice.

## 2. Stronger and more general versions.

We will first prove the following theorem.

THEOREM 2. *In an n-cube with a specified vertex v, the following are true:*

(i) *If* $2^n$ *pebbles are assigned to vertices of the n-cube, one pebble can be moved to v.*

(ii) *Let q be the number of vertices that are assigned an odd number of pebbles. If there are all together more than* $2^{n+1} - q$ *pebbles, then two pebbles can be moved to v.*

*Proof.* The proof is by induction on $n$. It is trivially true for $n = 0$. Suppose it is true for $n' < n$. The $n$-cube can be partitioned into two $(n-1)$-cubes, say $M_1$ and $M_2$, where $v$ is in $M_1$. Let $v'$ denote the vertex in $M_2$ adjacent to $v$. The edges between $M_1$ and $M_2$ form a perfect matching. Suppose $M_i$ contains $p_i$ pebbles with $q_i$ vertices having an odd number of pebbles, for $i = 1, 2$.

Suppose there are $p \geqq 2^n$ pebbles assigned to vertices of the $n$-cube. We will first show (i) holds. If $p_1 \geqq 2^{n-1}$, then by induction, in $M_1$, one pebble can be moved to $v$. We may assume $p_1 < 2^{n-1}$ and we consider the following two cases.

*Case* (a1). $q_2 > p_1$.

Since $p_2 = p - p_1 > 2^n - q_2$, by induction from (ii) in $M_2$ two pebbles can be moved to $v'$. Therefore one pebble can be moved to $v$.

*Case* (a2). $q_2 \leqq p_1$.

We apply pebbling steps to all vertices in $M_2$ and we can move at least $(p_2 - q_2)/2$ pebbles to vertices of $M_1$. Therefore, in $M_1$ we have all together $p_1 + (p_2 - q_2)/2 \geqq p_1 + (p_2 - p_1)/2 = (p_1 + p_2)/2 = 2^{n-1}$ pebbles. By induction, we can then move one pebble to $v$. This establishes (i).

It suffices now to prove (ii). Suppose there are $p = p_1 + p_2 > 2^{n+1} - q_1 - q_2$ pebbles assigned to vertices of the $n$-cube. We want to show that two pebbles can be moved to $v$. We consider the following three possibilities:

*Case* (b1). $p_1 > 2^n - q_1$.

By induction from (ii) in $M_1$, two pebbles can be moved to $v$.

*Case* (b2). $2^n - q_1 \geqq p_1 \geqq 2^{n-1}$.

Since $p_1 \geqq 2^{n-1}$, by induction from (i) in $M_1$ one pebble can be moved to $v$. Since $p_2 = p - p_1 > 2^{n+1} - q_1 - q_2 - p_1 \geqq 2^n - q_2$, two pebbles can be moved to $v'$ using induction from (ii) in $M_2$. Therefore an additional pebbling step results in moving one more pebble to $v$.

*Case* (b3). $p_1 < 2^{n-1}$.

For any integer $t$ satisfying $p_2 \geqq q_2 + 2t$, $t$ pebbles can be moved to vertices of $M_1$ while $p_2 - 2t$ pebbles remain in $M_2$. Note $p_2 > 2^{n+1} - q - p_1 = (2^n - q_2) + (2^n - q_1 - p_1) \geqq q_2 + (2^n - q_1 - p_1)$, where the last inequality follows since $q_2$ is at most $2^{n-1}$. Thus taking $t$ to be $2^{n-1} - \lceil (q_1 + p_1)/2 \rceil$, $t$ pebbles can be moved to $M_1$ leaving more than $2^n - q_2$ pebbles in $M_2$. In $M_1$ there are $p_1 + 2^{n-1} - \lceil (q_1 + p_1)/2 \rceil = 2^{n-1} + \lfloor (p_1 - q_1)/2 \rfloor \geqq 2^{n-1}$ pebbles. We can then move one pebble to $v$ in $M_1$ and at the same time move two pebbles to $v'$ in $M_2$, which will result in one additional pebble to $v$.

This completes the proof of Theorem 2.

We remark that Theorem 2 provides an efficient algorithm for pebbling in the $n$-cube. Furthermore in all the pebbling steps, pebbles are moved toward the specified vertex.

Theorem 2 can be slightly strengthened in Theorem 2′. This variation is useful for proving several generalized versions.

We say $i$ pebbles are extracted from an $n$-cube if $i$ pebbling steps are performed in a way that $2i$ pebbles are removed, but the placement of the $i$ new pebbles to the neighbors will be suspended until a later time.

THEOREM 2′. *If $2^{n+1} - r + w + 1$ pebbles are assigned to vertices of an n-cube while r vertices have at least one pebble, two pebbles can be moved to v after $\lfloor w/2 \rfloor$ pebbles are extracted from the cube.*

*Proof.* Again we will prove by induction on $n$ and view the $n$-cube as the union of two $(n-1)$-cubes $M_1$ and $M_2$ as described in the proof of Theorem 2, where $M_i$ contains $p_i$ pebbles with $r_i$ vertices having at least one pebble for $i = 1, 2$. Suppose $p = p_1 + p_2 = 2^{n+1} - r_1 - r_2 + w + 1$ pebbles are assigned to vertices of the $n$-cube.

We consider the following three possibilities:

*Case* (c1). $p_1 > 2^n - r_1$.

By induction in $M_1$, two pebbles can be moved to $v$ after $\lfloor (p_1 - 2^n + r_1 - 1)/2 \rfloor$ pebbles are extracted from $M_1$. In $M_2$, $\lceil (p_2 - r_2)/2 \rceil$ pebbles can be extracted. Therefore, altogether two pebbles can be moved to $v$ after $\lfloor (p_1 - 2^n + r_1 - 1 + p_2 - r_2)/2 \rfloor \geqq \lfloor w/2 \rfloor$ pebbles are extracted.

*Case* (c2). $2^n - r_1 \geqq p_1 \geqq 2^{n-1}$.

Since $p_1 \geqq 2^{n-1}$, one pebble can be moved to $v$ in $M_1$. Since $p_2 = p - p_1 > 2^{n+1} - r_1 - r_2 - p_1 + w \geqq 2^n - r_2 + w$, two pebbles can be moved to $v'$ after $\lfloor w/2 \rfloor$ pebbles are extracted by induction in $M_2$. Therefore one pebble can be moved to $v$.

*Case* (c3). $p_1 < 2^{n-1}$.

Since $p_2 > 2^{n+1} - r_1 - r_2 - p_1 + w = (2^n - r_2) + (2^n - r_1 - p_1) + w$, by induction two pebbles can be moved to $v'$ after $\lfloor (2^n - r_1 - p_1 + w)/2 \rfloor$ pebbles are extracted from $M_2$, among which $(2^n - r_1 - p_1)/2$ pebbles will be placed in $M_1$. In $M_1$ there are $p_1 + 2^{n-1} - \lceil (r_1 + p_1)/2 \rceil = 2^{n-1} + \lfloor (p_1 - r_1)/2 \rfloor \geqq 2^{n-1}$ pebbles. We can then move one pebble to $v$ in $M_1$ and at the same time move two pebbles to $v'$ in $M_2$, which will result in one additional pebble to $v$ after $\lfloor w/2 \rfloor$ pebbles are extracted from the cube.

This completes the proof of Theorem 2′.

One of the original versions of the problem proposed by Lagarias is the following theorem.

THEOREM 3. *For integers $p_1, p_2, \cdots, p_n \geqq 2$, suppose $p_1 p_2 \cdots p_n$ pebbles are assigned to the vertices of an n-cube $Q_n$. Each admissible pebbling step is to remove $p_i$ pebbles from a vertex $(a_1, \cdots, a_{i-1}, 1, a_{i+1}, \cdots, a_n)$ and place one pebble on $(a_1, \cdots, a_{i-1}, 0, a_{i+1}, \cdots, a_n)$. One can now repeatedly use the admissible pebbling steps to place one pebble on $(0, 0, \cdots, 0)$.*

*Proof.* The proof is very similar to that of Theorem 2 except that (ii) should read somewhat differently. We say an admissible step is of direction $i$ and cost $p_i$ if $p_i$ pebbles are removed from a vertex $(a_1, \cdots, a_{i-1}, 1, a_{i+1}, \cdots, a_n)$ and place one pebble on $(a_1, \cdots, a_{i-1}, 0, a_{i+1}, \cdots, a_n)$. For a fixed constant $k$, an admissible step of direction 0 if $k$ pebbles are removed from a vertex and one pebble will be kept to be placed later in a (possible future) direction of cost $k$. Let $q$ be the number of vertices that are assigned at least one pebble. If there are altogether more than $kp_1 \cdots p_n - q + w$ pebbles, then $k$ pebbles can be moved to $v$ after $\lfloor w_0/k \rfloor$, $\lfloor w_1/p_1 \rfloor$, $\lfloor w_2/p_2 \rfloor$, $\cdots$ $\lfloor w_n/p_n \rfloor$, where $w_0 + w_1 + \cdots + w_n = w$, are extracted from the cube in the $i$th direction of cost $p_i$, for $i = 0, 1, \cdots, n$, respectively.

**3. The pebbling number.** For a graph $G$, we define the pebbling number $f(G)$ to be the smallest integer $m$ such that for any distribution of $m$ pebbles to the vertices of $G$, one pebble can be moved to a specified vertex. Theorem 2 states that $f(Q_n) = 2^n$. We here include some facts about $f(G)$, most of which are quite straightforward (the proofs are left for the reader).

FACT 1. *$f(G) \geqq |V(G)|$.*

FACT 2. *$f(G) \geqq 2^D$ where $D = D(G)$ is the diameter of $G$.*

FACT 3. *If $G'$ is a spanning subgraph of $G$, then $f(G') \geqq f(G)$.*

FACT 4. *For a path $P_{t+1}$ on $t+1$ vertices, $f(P_{t+1}) = 2^t$.*

FACT 5. *For a complete graph $K_t$, $f(K_t) = t$.*

A graph $H$ is said to be a retract of a graph $G$ if there is a mapping from $V(G)$ to $V(H)$ which preserves edges, i.e., which maps adjacent vertices in $G$ to adjacent vertices in $H$. The reader is referred to [1], [3], [6] for various facts about retracts. The following simple fact turns out to be very useful.

FACT 6. *If $H$ is a retract of $G$, then $f(H) \leqq f(G)$.*

Duffus and Rival [2] showed that a finite distributive lattice of height $n$ is a retract of the $n$-dimensional cube. By using Theorem 3, we immediately have the following:

THEOREM 4. *Suppose $S$ is a partially ordered set in which each element is assigned an integer weight. In the corresponding finite distributive lattice, an admissible move is to remove $w$ pebbles from a vertex $v$ and place one pebble on $u$ where $w$ is the weight of the element in $v - u$. If $\prod_{x \in S} w(x)$ pebbles are assigned to vertices in the distributive lattice, then by repeatedly applying the admissible steps, one pebble can be moved to the empty set in the lattice.*

We remark that when $S$ consists on $n$ pairwise incomparable points, the distributive lattice is exactly the $n$-cube. Theorems 1–3 are all special cases of Theorem 4.

For any two graphs $G_1$ and $G_2$, we define the product $G_1 \square G_2$ to be the graph with vertex set $= \{(v_1, v_2) : v_1 \in V(G_1), v_2 \in V(G_2)\}$ and there is an edge between $(v_1, v_2)$ and $(v_1', v_2')$ if and only if ($v_1 = v_1'$ and $\{v_2, v_2'\} \in E(G_2)$) or ($\{v_1, v_1'\} \in E(G_1)$ and $v_2 = v_2'$). It is easy to see that the 1-cube is $K_2$; the 2-cube is $K_2 \square K_2$; and the $n$-cube $Q_n$ is $Q_{n-1} \square K_2$.

We say a graph $G$ satisfies the 2-pebbling property if two pebbles can be moved to a specified vertex when the total starting number of pebbles is $2f(G) - q + 1$, where $q$ is the number of vertices with at least one pebble. Clearly the $n$-cube satisfies the 2-pebbling property and the paths also have the 2-pebbling property.

THEOREM 5. *Suppose $G$ satisfies the 2-pebbling property. Then the following holds:*

(i) *$f(G \square K_t) \leqq tf(G)$.*

(ii) If *$f(G \square K_t) = tf(G)$, $G \square K_t$ satisfies the 2-pebbling property.*

The proof of Theorem 5 is extremely similar to the proof of Theorem 2 and 2′ and will be omitted here.

Using Theorem 5 together with Fact 1 yields the pebbling number for all products of cliques.

FACT 7. *$f(K_{t_1} \square K_{t_2} \square \cdots \square K_{t_s}) = t_1 t_2 \cdots t_s$.*

FACT 8. *$f(P_{t_1+1} \square P_{t_2+1} \square \cdots \square P_{t_s+1}) = 2^{t_1+t_2+\cdots t_s}$.*

*Proof.* On one hand, we have $f(P_{t_1+1} \square P_{t_2+1} \square \cdots P_{t_s+1}) \geqq 2^{t_1+\cdots+t_s}$ since the diameter is $t_1 + t_2 + \cdots + t_s$. On the other hand, since $P_{t_1+1} \square P_{t_2+1} \square \cdots \square P_{t_s+1}$ is a retract of the $(t_1 + t_2 + \cdots + t_s)$-cube, Fact 8 follows from Fact 6.

FACT 9. *For integers $p_1, p_2, \cdots, p_n \geqq 2$, $\alpha_1, \alpha_2, \cdots, \alpha_n \geqq 1$. Suppose $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$ pebbles are assigned to the vertices of $P_{\alpha_1+1} \square P_{\alpha_2+1} \square \cdots \square P_{\alpha_n+1}$. Each admissible pebbling step is to remove $p_i$ pebbles from a vertex*

$$(a_1, \cdots, a_{i-1}, x, a_{i+1}, \cdots, a_n)$$

*and place one pebble on $(a_1, \cdots, a_{i-1}, x-1, a_{i+1}, \cdots, a_n)$. One can repeatedly apply the admissible pebbling step to move a pebble to $(0, \cdots, 0)$.*

Fact 9 follows from Fact 6 and Theorem 3. We can now use Fact 9 to give a different proof to the result of Lemke and Kleitman [5]. That Theorem 3 implies Theorem 6 was Lagarias' motivation for formulating the problem.

THEOREM 6. *For any given integers $a_1, a_2, \cdots, a_d$, there is a nonempty subset $X \subseteq \{1, \cdots, d\}$ such that $d \mid \sum_{i\in X} a_i$ and $\sum_{i\in X} gcd(a_i, d) \leqq d$.*

*Proof.* Suppose $d = p_1^{\alpha_1}p_2^{\alpha_2}\cdots p_n^{\alpha_n}$. We consider $P_{\alpha_1+1} \square P_{\alpha_2+1} \square \cdots \square P_{\alpha_n+1}$. For each integer $a_i$, we place a pebble at $(b_1, \cdots, b_n)$ where $d/gcd(a_i, d) = p_1^{b_1}p_2^{b_2}\cdots p_n^{b_n}$. Suppose there are $p_i$ pebbles corresponding to numbers $x_1, \cdots, x_{p_i}$ at the vertex $(b_1, \cdots, b_n)$. There is a subset $S \subseteq \{1, \cdots, p_i\}$ such that $p_i \mid \sum_{i\in S} x_i = y$. We note that

$$\sum_{i\in S} gcd(x_i,d) \leqq p_i \cdot gcd(x_1,d)$$

$$= gcd(y,d).$$

We can then replace numbers $x_1, \cdots, x_{p_i}$ by the number $y$. If we can repeat this process and eventually move a pebble to $(0, \cdots, 0)$, then this implies that there is a subset $X \subseteq \{1, \cdots, d\}$ such that $d \mid \sum_{i\in X} a_i$ and $\sum_{i\in X} gcd(a_i, d) \leqq d$.

This completes the proof of Theorem 6.

Suppose $T$ is a tree with a specified vertex. $T$ can be viewed as a directed tree denoted by $T_v^*$ with edges directed toward the specified vertex $v$, also called the root. A path partition is a set of nonoverlapping directed paths the union of which is $T$. A path-partition is said to majorize another if the nonincreasing sequence of the path size majorizes that of the other. (That is $(a_1, a_2, \cdots, a_t) > (b_1, b_2, \cdots, b_t)$ if and only if $a_i > b_i$ where $i = \min\{j : a_j \neq b_j\}$.) A path-partition of a tree $T$ is said to be maximum if it majorizes all other path-partitions.

We define the pebbling number $f(T, v)$ to be the smallest integer $m$ such that if $m$ pebbles are assigned to the vertices of $T$, then one pebble can be moved to $v$.

FACT 10. *The pebbling number $f(T, v)$ for a vertex $v$ in a tree $T$ is $2^{a_1} + 2^{a_2} + \cdots + 2^{a_t} - t + 1$ where $a_1, a_2, \cdots, a_t$ is the sequence of the path size (i.e., the number of vertices in the path) in a maximum path-partition of $T_v^*$.*

Fact 10 is a special case of Fact 11 that considers the following general formulation. Let $f_k(T, v)$ denote the smallest integer $m$ such that if $m$ pebbles are assigned to the vertices of $T$ then $k$ pebbles can be moved to $v$.

FACT 11. *The pebbling number $f_k(T, v)$ for a vertex $v$ in a tree $T$ is $k2^{a_1} + 2^{a_2} + \cdots + 2^{a_t} - t + 1$ where $a_1, a_2, \cdots, a_t$ is the sequence of the path size in a maximum path-partition of $T_v^*$.*

*Proof.* The proof is by induction on the number of vertices of $T$. If we remove $v$ from $T$, the resulting graph is the union of subtrees $T_1, T_2, \cdots, T_s$ where $T_i$ contains a neighbor of $v$, say $u_i$. It is easy to see that for any $\lfloor k_1/2 \rfloor + \cdots + \lfloor k_s/2 \rfloor < k$ we have

$$f_k(T,v) - 1 \geqq f_{k_1+1}(T_1,u_1) + f_{k_2+1}(T_2,u_2) + \cdots + f_{k_s+1}(T_s,u_s) - s.$$

In fact,

$$f_k(T,v) - 1 = \operatorname*{Max}_{k_i} \{f_{k_1+1}(T_1,u_1) + f_{k_2+1}(T_2,u_2) + \cdots + f_{k_s+1}(T_s,u_s) - s\}.$$

Using the fact that $2^a + 2^b \geqq 2^{a-1} + 2^{b+1}$ if $a > b$, the maximum is achieved when $k_1 = k_3 = \cdots = k_s = 1$, $k_1 = 2k - 1$ while $T_1$ contains a vertex furthest from $v$. It is then straightforward to check that $f_k(T, v)$ has the desired expression.

FACT 12. *A tree satisfies the 2-pebbling property.*

*Proof.* From Fact 11 we know that $f_2(T, v) = f(T, v) + 2^{a_1}$ where $a_1$ is the number of edges in a longest directed path in $T$ with root $v$. It remains to show that

$$f(T,v) - |V(T)| + 1 \geqq 2^{a_1}$$

which follows from Fact 11.

**4. Questions on the pebbling number.** There are many problems on the pebbling number that we will mention here.

*Question* 1. Is it true that $f(G) = \max\{2^{D(G)}, |V(G)|\}$?

*Answer*. False. Consider the star of 3 edges. The pebbling number is 5 while $2^D = 4 = |V(G)|$.

*Question* 2. Is it true that $f(G_1 \square G_2) = f(G_1)f(G_2)$?

*Answer*. False. Consider $G_1 = K_3$ and $G_2 = P_3$.

$$f(K_3 \square P_3) = 9 \neq f(K_3)\cdot f(P_3) = 12.$$

There are many questions not resolved at this point.

*Question* 3 (RLG). Is it true that $f(G_1 \square G_2) \leqq f(G_1)f(G_2)$?

*Question* 4. Is it true that any graph has the 2-pebbling property?

If these two properties are true, the proof of Theorem 2 can be much simplified. Recently, Lemke constructed a counterexample to Question 4. His example does not provide a "no" answer to Question 3.

We remark that Theorem 5 can be used to determine $f(G)$ for a variety of graphs other than products of cliques or paths. For example, for the 5-cycle $C_5$, it is easy to see that $f(C_5) = 5$. Theorem 5 asserts $f(K_5 \square C_5) = 25$. It would be of interest to determine $f(C_5 \square C_5 \square \cdots \square C_5)$.

*Question* 5. Is it true that

$$f(\overbrace{C_5 \square C_5 \square \cdots \square C_5}^{nC_5\text{'s}}) = 5^n?$$

The following generalization of Theorem 6 was conjectured in [5].

CONJECTURE. *Any sequence of $|G|$ elements (not necessarily distinct) of the finite group $G$ contains a nonempty subsequence $g_1, g_2, \cdots, g_k$ such that $g_1 g_2 \cdots g_k = e$ and $\sum_{i=1}^{k} (1)/|g_i| \leqq 1$.*

When $G$ is cyclic, the conjecture is true as seen from Theorem 6 and [5].

**Acknowledgment.** The author would like to thank Noga Alon, Ron Graham, Dan Kleitman, and Mike Saks for many helpful comments. The author is also indebted to Joe Gallian and David Moews for their excellent comments and suggestions.

## REFERENCES

[1] H. J. BANDELT, *Retracts of hypercubes*, J. Graph Theory, 8 (1984), pp. 501–510.

[2] D. DUFFUS AND I. RIVAL, *Graphs orientable as distributive lattices*, Proc. Amer. Math. Soc., 88 (1983), pp. 197–200.

[3] P. HELL, *Graph retractions*, Colloq. Intern. Teorie Combinatorie II, Roma, 1976, pp. 263–268.

[4] J. LAGARIAS, *personal communication*, (1987).

[5] P. LEMKE AND D. KLEITMAN, *An addition theorem on the integers modulo n*, J. Number Theory, 31 (1989), pp. 335–345.

[6] H. M. MULDER, *n-cubes and median graphs*, J. Graph Theory, 4 (1980), pp. 107–110.

[7] M. SAKS, *personal communication*, (1987).

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 473–487, November 1989

# COMPLEXITY OF SCHEDULING PARALLEL TASK SYSTEMS*

JIANZHONG DU† AND JOSEPH Y-T. LEUNG†

**Abstract.** One of the assumptions made in classical scheduling theory is that a task is always executed by one processor at a time. With the advances in parallel algorithms, this assumption may not be valid for future task systems. In this paper, a new model of task systems is studied, the so-called Parallel Task System, in which a task can be executed by one or more processors at the same time. The complexity of scheduling Parallel Task Systems to minimize the schedule length is examined. For nonpreemptive scheduling, it is shown that the problem is strongly NP-hard even for two processors when the precedence constraints consist of a set of chains. For independent tasks, the problem is strongly NP-hard for five processors, but solvable in pseudo-polynomial time for two and three processors. For preemptive scheduling, it is shown that the problem is strongly NP-hard for arbitrary number of processors for a set of independent tasks. Furthermore, it is shown that it is NP-hard, but solvable in pseudo-polynomial time, for a fixed number of processors.

**Key words.** NP-hard, pseudo-polynomial time, parallel task system, nonpreemptive scheduling, preemptive scheduling, schedule length

**AMS(MOS) subject classifications.** 90B30, 68R05

## 1. Introduction.

One of the assumptions made in classical scheduling theory [2] is that a task is always executed by one processor at a time. With the advances in parallel algorithms, this assumption may not be valid for future task systems. In this paper, we propose a new model of task systems, the so-called Parallel Task System (PTS), to include tasks that implement parallel algorithms. We hypothesize that a task can be executed by $1, 2, \cdots, m$ processors, each with different execution time requirements. In scheduling a parallel task, we can assign any number of processors to it. However, once the number of processors assigned to a task is determined, it will remain fixed throughout the execution of the task. This assumption is reasonable since most parallel algorithms are implemented in such a way that it is difficult to dynamically change the number of processors assigned to them. For those parallel algorithms which have different processor requirements during different parts of their programs, we can model them by a chain of parallel tasks such that each task in the chain has the same processor requirement. The main purpose of this paper is to examine the complexity of scheduling a Parallel Task System on $m \geqq 2$ identical processors to minimize the schedule length. Our study includes both nonpreemptive and preemptive scheduling disciplines.

Formally, we are given a set $P = \{P_1, P_2, \cdots, P_m\}$ of $m \geqq 2$ identical processors and a Parallel Task System PTS $= (TS, \tau, G)$, where $TS = \{T_1, T_2, \cdots, T_n\}$ is a set of $n$ parallel tasks, $\tau$ is a function giving the execution times of a parallel task ($\tau(T_i, j)$ is the execution time of $T_i$ when executed simultaneously by $j$ processors, $1 \leqq j \leqq m$), and $G = (TS, E)$ is a directed acyclic graph describing the precedence constraints among the parallel tasks. Since a task takes less time to run on more processors, we assume $\tau(T_i, j) \geqq \tau(T_i, j+1)$ for each $1 \leqq i \leqq n$ and $1 \leqq j < m$. Our goal is to schedule PTS on the $m$ processors such that the length of the schedule is minimized. Such a schedule will be called an optimal schedule.

Our model is similar to the one proposed and studied in [1]. The only difference between the two models is that in the model of Blazewicz, Drabowski, and Welgarz, a task must be executed simultaneously by a specified number of processors, whereas in

* Received by the editors January 19, 1989; accepted for publication April 4, 1989.

† Computer Science Program, University of Texas at Dallas, Richardson, Texas 75080. This research was supported in part by Office of Naval Research grant N00014-87-K-0833.

our model, it can be executed by any number not exceeding $m$. For convenience, we shall call their model the Multiprocessor Task System (MTS). We can model a Multiprocessor Task System by a Parallel Task System as follows. Let $T_i$ be a task in a Multiprocessor Task System that requires $k$ processors, and let $p_i$ be the execution time of $T_i$. When transformed into a Parallel Task System, we can let $\tau(T_i, j) = \infty$ for $1 \leqq j < k$, and $\tau(T_i, j) = p_i$ for $k \leqq j \leqq m$. Clearly, an optimal schedule for the Multiprocessor Task System has the same length as an optimal schedule for the transformed Parallel Task System. We shall denote a Multiprocessor Task System by MTS = $(TS, \tau, G)$. Since a Multiprocessor Task System can be transformed into a Parallel Task System, it is easy to see that if a special case of finding an optimal schedule for Multiprocessor Task Systems is NP-hard (or strongly NP-hard), then the corresponding case for Parallel Task Systems is also NP-hard (or strongly NP-hard). Conversely, if a special case of finding an optimal schedule for Parallel Task Systems is solvable in polynomial time (or pseudo-polynomial time), then the corresponding case for Multiprocessor Task Systems is also solvable in polynomial time (or pseudo-polynomial time). Note that the above two implications do not hold if the special case contains the condition that the tasks have equal execution times.

In [1], it has been shown that finding an optimal nonpreemptive schedule for a Multiprocessor Task System with equal execution times and empty precedence constraints is strongly NP-hard for arbitrary $m$. However, the problem can be solved in polynomial time for each fixed $m$ [1]. It is also shown in [1] that finding an optimal preemptive schedule for a Multiprocessor Task System with empty precedence constraints is solvable in polynomial time for each fixed $m$.

It is easy to see that a task system defined in classical scheduling theory can also be transformed into a Parallel Task System. We shall call such a task system a Classical Task System and denote it by CTS = $(TS, \tau, G)$. In the literature, there are many complexity results concerning the Classical Task Systems [2], [3]. We shall mention a couple of these results that are relevant to this paper. Finding an optimal nonpreemptive schedule for a Classical Task System with empty precedence constraints is NP-hard, but solvable in pseudo-polynomial time, for each $m \geqq 2$ [3]. Finding an optimal preemptive schedule for the same problem can be solved in polynomial time even when $m$ is arbitrary [6].

In this paper, we show that finding an optimal nonpreemptive schedule for a Parallel Task System with the precedence constraints consisting of chains is strongly NP-hard for each $m \geqq 2$. When the precedence constraints are empty, the problem can be solved in pseudo-polynomial time for $m = 2$ and 3, and it becomes strongly NP-hard for each $m \geqq 5$. For $m = 4$, it is not known whether the problem is strongly NP-hard or solvable in pseudo-polynomial time. For preemptive scheduling, we show that finding an optimal schedule for a Parallel Task System with empty precedence constraints is NP-hard, but solvable in pseudo-polynomial time, for each $m \geqq 2$. For arbitrary $m$, we show that the problem is strongly NP-hard.

We now define some notations and conventions that are used consistently in the remainder of this paper. Let $S$ be a schedule of a Parallel Task System, PTS = $(TS, \tau, G)$. We say that task $T_i$ is a $k$-processor task with respect to $S$ if $T_i$ is executed by $k$ processors in $S$. A lower bound of the schedule length of $S$ is given by $(1/m) \sum_{i=1}^{n} \min_{j=1}^{m} \{j\tau(T_i, j)\}$. This follows from the observation that $\min_{j=1}^{m} \{j\tau(T_i, j)\}$ is the minimum processing time requirement to execute task $T_i$. If MTS = $(TS, \tau, G)$ is a Multiprocessor Task System, we say that task $T_i$ is a $k$-processor task if $T_i$ requires $k$ processors to execute. The NP-hardness results in this paper are shown by reducing the PARTITION problem and the 3-PARTITION problem to our problems. The PARTITION problem is known to be NP-complete [3] and the 3-PARTITION problem is known to be strongly NP-complete [3].

PARTITION problem. Given a list $A = (a_1, a_2, \cdots, a_z)$ of $z$ integers such that $\sum_{i=1}^{z} a_i = B$, can $I = \{1, 2, \cdots, z\}$ be partitioned into $I_1$ and $I_2$ such that $\sum_{i\in I_1} a_i = \sum_{i\in I_2} a_i = B/2$?

3-PARTITION problem. Given a list $A = (a_1, a_2, \cdots, a_{3z})$ of $3z$ integers such that $\sum_{i=1}^{3z} a_i = zB$ and $B/4 < a_i < B/2$ for each $1 \leqq i \leqq 3z$, can $I = \{1, 2, \cdots, 3z\}$ be partitioned into $I_1, I_2, \cdots, I_z$ such that $\sum_{i\in I_j} a_i = B$ for each $1 \leqq j \leqq z$?

The organization of the paper is as follows. In § 2, we shall study the complexity of nonpreemptive scheduling of Parallel Task Systems. In § 3, we study the complexity of preemptive scheduling. Finally, we draw some concluding remarks in the last section.

## 2. Nonpreemptive scheduling.

In this section, we consider the complexity of finding an optimal nonpreemptive schedule for a Parallel Task System PTS = $(TS, \tau, G)$. When the precedence constraints are empty, the problem is easily seen to be NP-hard for every $m \geqq 2$. This result follows directly from the result in Classical Task Systems. Our interest here is in drawing a sharp boundary between ordinary and strong NP-hardness for the problem. We shall show that if the precedence constraints are not empty, then the problem is strongly NP-hard for every $m \geqq 2$. If the precedence constraints are empty, it remains strongly NP-hard for $m \geqq 5$, while for $m = 2$ and 3, it is solvable in pseudo-polynomial time. It is not known whether the problem is strongly NP-hard or solvable in pseudo-polynomial time for $m = 4$. In proving NP-hardness results in this section, we shall be showing the corresponding decision problem for Multiprocessor Task Systems to be NP-complete. From the discussions in § 1, the result immediately follows for Parallel Task Systems. Thus, we define the following decision problem.

*Problem* M1. Given $m$, $\omega$, and a Multiprocessor Task System MTS = $(TS, \tau, G)$, is there a nonpreemptive schedule of MTS on $m$ identical processors such that the schedule length is no larger than $\omega$?

THEOREM 1. *Problem* M1 *is strongly* NP-*complete for* $m = 2$ *and the precedence constraints consisting of a set of chains.*

*Proof.* We shall reduce the 3-PARTITION problem to Problem M1. Given an instance of the 3-PARTITION problem, $A = (a_1, a_2, \cdots, a_{3z})$, we construct an instance of Problem M1 as follows. Let $TS = \{Q_1, Q_2, \cdots, Q_{4z}\} \cup \{R_1, R_2, \cdots, R_{z-1}\}$, where $Q_i$, $1 \leqq i \leqq 4z$, is a 1-processor task, and $R_i$, $1 \leqq i \leqq z-1$, is a 2-processor task. $\tau(Q_i) = a_i$ for $1 \leqq i \leqq 3z$, $\tau(Q_i) = B$ for $3z+1 \leqq i \leqq 4z$, and $\tau(R_i) = 1$ for $1 \leqq i \leqq z-1$. $G$ is as shown in Fig. 1. Finally, we let $\omega = z(B+1) - 1$. It is easy to see that the construction can be done in polynomial time.

If the instance of the 3-PARTITION problem has a solution, then it is clear that the constructed instance of Problem M1 also has a solution. Conversely, if the constructed instance of Problem M1 has a solution, then the chain in $G$ must be continuously executing from time 0 to $\omega$ on one processor. Thus, on the other processor, there will be $z$ separated intervals, each of length $B$, left for tasks $Q_i$, $1 \leqq i \leqq 3z$. This implies there is a solution for the instance of the 3-PARTITION problem. □

The result of Theorem 1 motivates us to consider the case of empty precedence constraints. Unfortunately, this case is still strongly NP-hard for $m \geqq 5$, as will be shown in Theorem 2. First, we need to define the following operation. Suppose $P_x$ and $P_y$ are two different processors, and on each of them there is a task finishing at time $t$. Then a new schedule can be obtained by swapping all the tasks (or parts of the tasks) executing

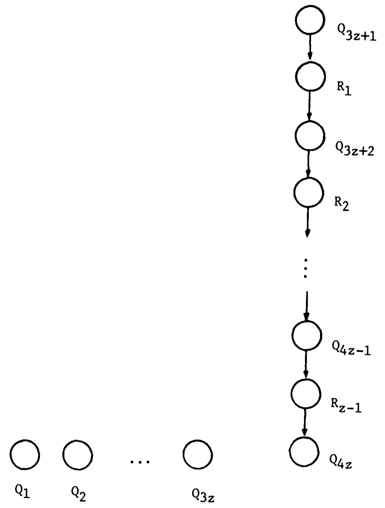


FIG. 1. *Reduction in Theorem 1.*

after time $t$ on $P_x$ with those on $P_y$. In the following discussions, we shall call this operation on a schedule a swapping operation, denoted by SWAP$(t, P_x, P_y)$. Observe that the swapping operation preserves the starting and finishing times of all tasks.

THEOREM 2. *Problem* M1 *is strongly* NP-*complete for* $m = 5$ *and empty precedence constraints.*

*Proof.* We reduce the 3-PARTITION problem to Problem M1. Let $A = (a_1, a_2, \cdots, a_{3z})$ be an instance of the 3-PARTITION problem. Without loss of generality, we may assume that $B > z(3z + 2)$. Otherwise, we can multiply each partition element by $z(3z + 2)$ without changing the solution of the problem. We construct an instance of Problem M1 as follows. Let $TS = Q \cup R \cup T$, where $Q = \{Q_1, Q_2, \cdots, Q_{3z+2}\}$ are the 1-processor tasks, $R = \{R_1, R_2, \cdots, R_{6z}\}$ are the 2-processor tasks, and $T = \{T_1, T_2, \cdots, T_{3z+1}\}$ are the 3-processor tasks. We divide $Q$ into three groups, $X_1 = \{Q_i | 1 \leqq i \leqq z + 1\}$, $X_2 = \{Q_i | z + 2 \leqq i \leqq 2z + 2\}$, and $X_3 = \{Q_i | 2z + 3 \leqq i \leqq 3z + 2\}$. $R$ is divided into two groups, $Y_1 = \{R_i | 1 \leqq i \leqq 3z\}$ and $Y_2 = \{R_i | 3z + 1 \leqq i \leqq 6z\}$. $T$ is divided into three groups, $Z_1 = \{T_i | 1 \leqq i \leqq z\}$, $Z_2 = \{T_i | z + 1 \leqq i \leqq 2z\}$, and $Z_3 = \{T_i | 2z + 1 \leqq i \leqq 3z + 1\}$. $\tau$ is defined as follows:

$$\tau(Q_i) = \begin{cases} B^2 + B^3 + (3z+1)B^5 & \text{if } 1 \leqq i \leqq z \\ B^2 + zB^5 & \text{if } i = z+1 \\ B^2 + 3zB^5 & \text{if } i = z+2 \\ B^2 + B^4 + 3zB^5 & \text{if } z+3 \leqq i \leqq 2z+1 \\ B^2 + B^4 + 2zB^5 & \text{if } i = 2z+2 \\ B^3 + B^4 + (3z+2)B^5 & \text{if } 2z+3 \leqq i \leqq 3z+2. \end{cases}$$

$$\tau(R_i) = \begin{cases} (3z - 2\lfloor i/3 \rfloor)B^5 - B & \text{if } 1 \leqq i \leqq 3z \text{ and } i \bmod 3 = 1 \\ \lceil i/3 \rceil B^5 & \text{if } 1 \leqq i \leqq 3z \text{ and } i \bmod 3 = 0 \text{ or } 2 \\ a_{i-3z} & \text{if } 3z + 1 \leqq i \leqq 6z. \end{cases}$$

$$\tau(T_i) = \begin{cases} B^4 \text{ if } 1 \leqq i \leqq z \\ B^3 \text{ if } z+1 \leqq i \leqq 2z \\ B^2 \text{ if } 2z+1 \leqq i \leqq 3z+1. \end{cases}$$

Finally, we let $\omega = (z+1)B^2 + zB^3 + zB^4 + z(3z+2)B^5$. It is easy to see that the construction can be done in polynomial time.

If the given instance of the 3-PARTITION problem has a solution, then there exist $z$ sets $\{I_1, I_2, \cdots, I_z\}$ such that $I_1 \cup I_2 \cup \cdots \cup I_z = \{1, 2, \cdots, 3z\}$ and $\sum_{i \in I_j} a_i = B$ for each $1 \leqq j \leqq z$. It is easy to verify that the schedule shown in Fig. 2 is a valid schedule for $TS$ on five processors with a length $\omega$. Conversely, suppose that a schedule, say $S$, of $TS$ has a length $\omega$. Dictated by the total processing time requirement of all tasks, we observe that $S$ cannot have any idle processor time before $\omega$ and it must have a length exactly $\omega$. To show that the instance of the 3-PARTITION problem has a solution, we shall show that by a finite number of swapping operations, SWAP$(t, P_x, P_y)$, $S$ can be transformed into a schedule isomorphic to the schedule shown in Fig. 2. This is accomplished by proving the following three claims.

*Claim* 1. $X_1$, $X_2$, and $X_3$ is the unique partition of $Q$ into three subsets such that the total execution time of the tasks in each subset is no larger than $\omega$.

*Proof.* Obviously, $X_1$, $X_2$, and $X_3$ are such a partition of $Q$. We need to show the uniqueness. Let us consider just the coefficients of the $B^5$ term in the expressions of the execution times of the tasks in $Q$. Observe that their sum is exactly $3z(3z+2)$. Since $B > z(3z+2)$, we have $\omega < (1 + z(3z+2))B^5$. Therefore, since the coefficient of the $B^5$ term in the expression of $\omega$ is $z(3z+2)$, any partition of $Q$ into three subsets such that the total execution time of the tasks in each subset is no larger than $\omega$ is also one such that the sum of the coefficients of the $B^5$ term in the expressions of the execution times of the tasks in each subset is equal to $z(3z+2)$. Thus, instead of considering the uniqueness problem, we first answer a more specific question: Given $3z+2$ integers with $z$ of them being $3z$, $z$ of them being $3z+1$, $z$ of them being $3z+2$, one of them being $2z$, and one of them being $z$, is the partition of them into three subsets, each totalling $z(3z+2)$, unique? In fact, it is readily shown that $(z, 0, 0, 1, 0)$, $(x, z-2x, x, 0, 1)$ for $0 \leqq x \leqq z/2$, and $(0, 0, z, 0, 0)$ are the only integer solutions to the equation

$$3zx_1 + (3z+1)x_2 + (3z+2)x_3 + 2zx_4 + zx_5 = z(3z+2),$$

subject to $0 \leqq x_1, x_2, x_3 \leqq z$, and $0 \leqq x_4, x_5 \leqq 1$. Hence, it is clear that the answer to our question is positive, and so reasoning backwards, we have proved that $X_1$, $X_2$, and $X_3$ is the unique partition. $\square$

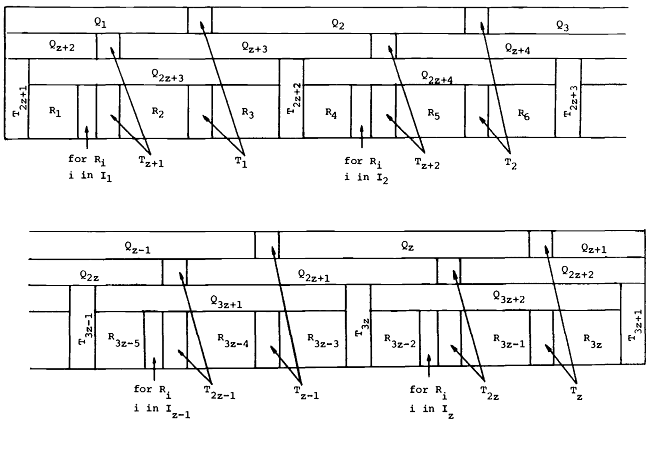


FIG. 2. *The optimal schedule in Theorem 2.*

*Claim* 2. By a finite number of swapping operations, $S$ can be transformed into a schedule $\hat{S}$ with the same length such that $g(P_i) = X_i \cup Z_i$ for $1 \leqq i \leqq 3$, and $g(P_4) = g(P_5) = R \cup T$, where $g(P_i)$ denotes the set of tasks which use processor $P_i$ in $\hat{S}$.

*Proof.* First, we show that by a finite number of swapping operations, we can make all 3-processor tasks use processor $P_5$. If the first 3-processor task in $S$ does not use $P_5$, then by SWAP$(0, P_5, \hat{P})$, where $\hat{P}$ is one of the processors that it uses, we can make it so. Suppose that after a certain number of swapping operations, we have made the first $i$ 3-processor tasks all use $P_5$. Consider the $(i + 1)$st 3-processor task. Because $m = 5$, any two 3-processor tasks must use at least one common processor in any schedule. Let $\hat{P}$ be the common processor of the $i$th and the $(i + 1)$st 3-processor tasks, and let $t$ be the finishing time of the $i$th 3-processor task. Then if $\hat{P}$ is not the same as $P_5$, by one more swapping operation SWAP$(t, P_5, \hat{P})$, we would have made the first $i + 1$ 3-processor tasks all use $P_5$. Therefore, $S$ can be transformed such that all 3-processor tasks use $P_5$.

Let us denote the schedule transformed from $S$ as above by $\tilde{S}$. Note that $\tilde{S}$ has length $\omega$ and it has no internal idle processor time. Now, since all 3-processor tasks use $P_5$ in $\tilde{S}$, the total execution time of all other tasks which use $P_5$ has to be exactly $z(3z + 2)B^5$. Since $B > z(3z + 2)$ and since each 1-processor task contains at least one of $B^2$, $B^3$, and $B^4$ terms in the expression of its execution time, there can be no tasks from $Q$ executed by $P_5$. Thus, the only tasks that can execute on $P_5$ are from $R$. Since the total execution time of all tasks in $R$ is $z(3z + 2)B^5$, we immediately have $g(P_5) = R \cup T$.

Next, we show that $\tilde{S}$ can be transformed into $\hat{S}$ such that the tasks which use $P_5$ in $\hat{S}$ also use $P_4$. If the first task which uses $P_5$ in $\tilde{S}$ does not use $P_4$, then by SWAP$(0, P_4, \hat{P})$, where $\hat{P} \neq P_5$ is one of the other processors that it uses, we can make it so. Suppose that after a certain number of swapping operations, we have made the first $i$ tasks on $P_5$ also use $P_4$. Consider the $(i + 1)$st task on $P_5$. Let $\hat{P} \neq P_5$ be one of the other processors that the $(i + 1)$st task uses. Notice that the starting time of the $(i + 1)$st task is the same as the finishing time of the $i$th task, say time $t$. Thus, if $\hat{P}$ is not the same as $P_4$, then by SWAP$(t, P_4, \hat{P})$, we can make the $(i + 1)$st task also use $P_4$. Therefore, by a finite number of swapping operations, we can make the tasks which use $P_5$ also use $P_4$. Since the swapping operations do not involve $P_5$, we have $g(P_4) = g(P_5) = R \cup T$ in $\hat{S}$.

Finally, we need to show that in $\hat{S}$, $g(P_i) = X_i \cup Z_i$ for $1 \leqq i \leqq 3$. Observe that the only tasks left to execute on processors $P_1$, $P_2$, and $P_3$ are the tasks in $Q$. Using Claim 1 and by renaming processors if necessary, it is easy to see that the claim is true. □

*Claim* 3. The starting times of the 3-processor tasks in $\hat{S}$ are isomorphic to those in the schedule shown in Fig. 2, up to a renaming of the tasks with the same execution time and a switching of both ends of $\hat{S}$.

*Proof.* Let us first see at what time a 3-processor task can start in $\hat{S}$. Since $B > z(3z + 2)$ and since $g(P_4) = g(P_5) = R \cup T$, we know, from the form of the execution times of the 2-processor tasks, that the starting time of a 3-processor task $\hat{T}$ is of the form

$$x_1 + x_2B^2 + x_3B^3 + x_4B^4 + x_5B^5,$$

where $x_1$ is an integer with $|x_1| < B^2$, and $0 \leqq x_i < B$ is the number of 3-processor tasks in $Z_{5-i}$ which start before task $\hat{T}$, for each $2 \leqq i \leqq 4$. Furthermore, since by Claim 2 $g(P_i) = X_i \cup Z_i$ for each $1 \leqq i \leqq 3$, the following three statements must be true: (1) If $\hat{T}$ is in $Z_1$, then $x_1 = 0$ and either $x_2 = x_3$ or $x_2 = x_3 + 1$. (2) If $\hat{T}$ is in $Z_2$, then $x_1 = 0$ and either $x_2 = x_4$ or $x_2 = x_4 + 1$. (3) If $\hat{T}$ is in $Z_3$, then $x_1 = 0$ and $x_3 = x_4$. From the above facts and the fact that $g(P_4) = g(P_5)$ by Claim 2, it is easy to show that the

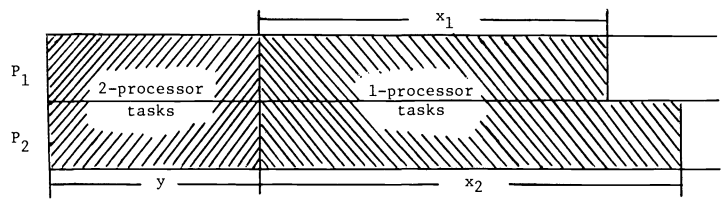


FIG. 3. *Canonical schedule for $m = 2$.*

3-processor tasks in $\hat{S}$ must interleave in the same manner as in Fig. 2, possibly with the two ends interchanged and the tasks with the same execution time renamed. □

By Claim 3, we may assume that the starting times of the 3-processor tasks in $\hat{S}$ are exactly the same as the schedule shown in Fig. 2. Hence, the distance between the $j$th and the $(j + 1)$st 3-processor tasks in $\hat{S}$ is $\tau(R_j) + B$ when $j \bmod 3 = 1$ and $\tau(R_j)$ when $j \bmod 3 = 0$ or 2. We now consider how the 2-processor tasks are scheduled on processors $P_4$ and $P_5$. Since $B > z(3z + 2)$, we see from the forms of the execution times of the 2-processor tasks that there must be one task in $Y_1$ between every pair of consecutive 3-processor tasks in order for $\hat{S}$ to have no internal idle processor time. Consequently, $z$ separated intervals of length $B$ are left over for the tasks in $Y_2$. Thus, we conclude there is a solution for the given instance of the 3-PARTITION problem. □

Theorem 2 shows that finding an optimal nonpreemptive schedule for a Parallel Task System with empty precedence constraints is strongly NP-hard for $m \geqq 5$. In the following, we shall show that the same problem can be solved in pseudo-polynomial time for $m = 2$ and 3. First, we need to prove the following lemma.

LEMMA 1. *Let* PTS = $(TS, \tau, G)$ *be a Parallel Task System with empty precedence constraints. Let $S$ and $\hat{S}$ be an optimal nonpreemptive schedule of* PTS *on* 2- *and* 3-*processors, respectively. Then, $S$ and $\hat{S}$ can be transformed, with no change in schedule length, into a canonical schedule as shown in Figs.* 3 *and* 4, *respectively.*

*Proof.* Let $S$ and $\hat{S}$ be an optimal nonpreemptive schedule of PTS on 2- and 3-processors, respectively. $S$ can be transformed into a canonical schedule shown in Fig. 3 as follows. Move all 2-processor tasks with respect to $S$ to the beginning of the schedule, and shift the 1-processor tasks with respect to $S$ towards the beginning of the schedule. It is clear that this transformation cannot change the length of the schedule.

$\hat{S}$ can be transformed into a canonical schedule as shown in Fig. 4 as follows. (1) Apply the swapping operations as in Claim 2 of Theorem 2 to make all 2-processor tasks with respect to $\hat{S}$ use processor $P_2$; (2) move all 3-processor tasks with respect to $\hat{S}$ to

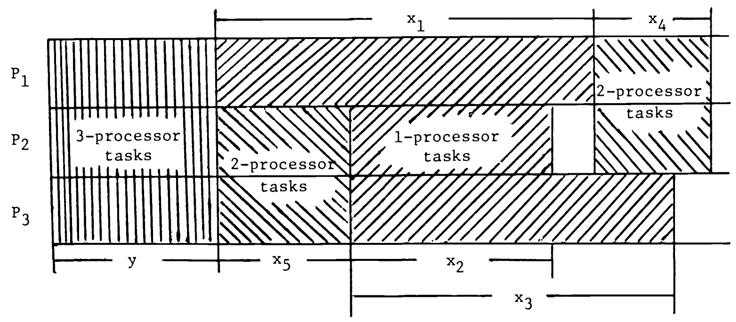


FIG. 4. *Canonical schedule for $m = 3$.*

the beginning of the schedule; (3) move the 2-processor tasks with respect to $\hat{S}$ which use processors $P_2$ and $P_3$ right next to the 3-processor tasks; (4) move the 2-processor tasks with respect to $\hat{S}$ which use processors $P_1$ and $P_2$ to the end of the schedule; and (5) shift all tasks towards the beginning of the schedule. It is easy to see that the schedule obtained is as shown in Fig. 4 and has the same length as $\hat{S}$. □

From Lemma 1, we see that an optimal nonpreemptive schedule, $S$, for 2-processors is completely determined by three numbers: the total execution times, $x_1$ and $x_2$, of the 1-processor tasks with respect to $S$ executed on processors $P_1$ and $P_2$, respectively, and the total execution time, $y$, of the 2-processor tasks with respect to $S$. An optimal nonpreemptive schedule, $\hat{S}$, for 3-processors is completely determined by six numbers: the total execution times, $x_1$, $x_2$, and $x_3$, of the 1-processor tasks with respect to $\hat{S}$ executed on processors $P_1$, $P_2$, and $P_3$, respectively, the total execution time, $x_4$, of the 2-processor tasks with respect to $\hat{S}$ which use $P_1$ and $P_2$, the total execution time, $x_5$, of the 2-processor tasks with respect to $\hat{S}$ which use $P_2$ and $P_3$, and the total execution time, $y$, of the 3-processor tasks with respect to $\hat{S}$. Using these characterizations, we can develop pseudo-polynomial time algorithms for $m = 2$ and 3.

Let PTS = $(TS, \tau, G)$ be a Parallel Task System with empty precedence constraints to be scheduled on two processors. Let $TS = \{T_1, T_2, \cdots, T_n\}$, and let $M = \sum_{i=1}^{n} \tau(T_i, 1)$. We construct a three-dimensional table $F(i, x_1, x_2)$, $0 \leqq i \leqq n$, $0 \leqq x_1 + x_2 \leqq M$, where $F(i, x_1, x_2)$ gives the smallest total execution time of the 2-processor tasks, among all schedules of the first $i$ tasks in $TS$, such that the total execution times of the 1-processor tasks executed on processors $P_1$ and $P_2$ are $x_1$ and $x_2$, respectively. The table can be computed as follows. (1) $F(0, 0, 0) = 0$ and $F(0, x_1, x_2) = \infty$ for $0 < x_1 + x_2 \leqq M$; (2) For each $1 \leqq i \leqq n$ and $0 \leqq x_1 + x_2 \leqq M$,

$$\begin{aligned} F(i,x_1,x_2) = \min\{&F(i-1,x_1,x_2)+\tau(T_i,2),\\ &F(i-1,x_1-\tau(T_i,1),x_2), F(i-1,x_1,x_2-\tau(T_i,1))\}. \end{aligned}$$

Note that the first term is obtained by making $T_i$ as a 2-processor task, and the second and the third terms are obtained by making $T_i$ as a 1-processor task executed on processors $P_1$ and $P_2$, respectively. Also, the second term will be dropped if $x_1 - \tau(T_i, 1) < 0$. Similarly, the third term will be dropped if $x_2 - \tau(T_i, 1) < 0$. After the table is constructed, we find $x_1^*$ and $x_2^*$ such that

$$\max\{x_1^*, x_2^*\} + F(n, x_1^*, x_2^*) = \min\{\max\{x_1, x_2\} + F(n, x_1, x_2)\}.$$

The optimal schedule length is given by $\max\{x_1^*, x_2^*\} + F(n, x_1^*, x_2^*)$. It is easy to see that the running time of the algorithm is $O(nM^2)$.

For $m = 3$, we can determine the optimal schedule length as follows. Let $M_1 = \sum_{i=1}^{n} \tau(T_i, 1)$ and $M_2 = \sum_{i=1}^{n} \tau(T_i, 2)$. We construct a six-dimensional table

$$F(i,x_1,x_2,x_3,x_4,x_5), 0 \leqq i \leqq n, 0 \leqq x_1+x_2+x_3 \leqq M_1, 0 \leqq x_4+x_5 \leqq M_2,$$

where $F(i, x_1, x_2, x_3, x_4, x_5)$ gives the smallest total execution time of the 3-processor tasks, among all schedules of the first $i$ tasks of $TS$, such that the total execution times of the 1-processor tasks executed on processors $P_1$, $P_2$, and $P_3$ are $x_1$, $x_2$, and $x_3$, respectively, the total execution time of the 2-processor tasks which use $P_1$ and $P_2$ is $x_4$, and the total execution time of the 2-processor tasks which use $P_2$ and $P_3$ is $x_5$. The table can be computed as follows. (1) $F(0, 0, 0, 0, 0, 0) = 0$ and $F(0, x_1, x_2, x_3, x_4, x_5) = \infty$ for $0 < x_1 + x_2 + x_3 \leqq M_1$ and $0 < x_4 + x_5 \leqq M_2$; (2) For $1 \leqq i \leqq n$, $0 \leqq x_1 + x_2 +$

$x_3 \leqq M_1$, and $0 \leqq x_4 + x_5 \leqq M_2$,

$$\begin{aligned} F(i,x_1,x_2,x_3,x_4,x_5) = \min\{&F(i-1,x_1,x_2,x_3,x_4,x_5)+\tau(T_i,3),\\ &F(i-1,x_1-\tau(T_i,1),x_2,x_3,x_4,x_5),\\ &F(i-1,x_1,x_2-\tau(T_i,1),x_3,x_4,x_5),\\ &F(i-1,x_1,x_2,x_3-\tau(T_i,1),x_4,x_5),\\ &F(i-1,x_1,x_2,x_3,x_4-\tau(T_i,2),x_5),\\ &F(i-1,x_1,x_2,x_3,x_4,x_5-\tau(T_i,2))\}. \end{aligned}$$

The optimal schedule length is given by

$$\begin{aligned} &\max\{x_1^* + x_4^*, x_2^* + x_4^* + x_5^*, x_3^* + x_5^*\} + F(n, x_1^*, x_2^*, x_3^*, x_4^*, x_5^*)\\ &\qquad = \min\{\max\{x_1+x_4, x_2+x_4+x_5, x_3+x_5\} + F(n,x_1,x_2,x_3,x_4,x_5)\}. \end{aligned}$$

The running time of the algorithm is $O(nM^5)$, where $M = \max\{M_1, M_2\}$.

From the previous discussions, we have the following theorem.

THEOREM 3. *The problem of finding an optimal nonpreemptive schedule for a Parallel Task System with empty precedence constraints is solvable in pseudo-polynomial time for $m = 2$ and 3.*

**3. Preemptive scheduling.** In this section, we examine the complexity of finding an optimal preemptive schedule for a Parallel Task System with empty precedence constraints. We first show that the problem is strongly NP-hard for arbitrary $m$. We then show that it is NP-hard, but solvable in pseudo-polynomial time, for every $m \geqq 2$. The following lemma is instrumental in proving the strong NP-hardness result.

LEMMA 2. *Suppose $f$ is a positive function such that $f(x) < f(y)$ if $x < y$, and $(c_1, c_2, \cdots, c_z)$ is a list of $z$ real numbers such that $c_1 + c_2 + \cdots + c_z = 0$. If $\sum_{i=1}^{j} c_i f(i) \geqq 0$ for each $1 \leqq j \leqq z$, then $c_i = 0$ for each $1 \leqq i \leqq z$.*

*Proof.* We prove the lemma by induction on $z$. It is obvious that the lemma holds for $z \leqq 2$, since $f$ is positive and strictly increasing. Suppose the lemma is true for $z = k$; we want to show it is true for $z = k + 1$. Let $f$ be a positive function and $(c_1, c_2, \cdots, c_{k+1})$ be a list of real numbers that satisfy the conditions of the lemma. Observe that $c_1 \geqq 0$, since $f$ is a positive function and $c_1 f(1) \geqq 0$. Let $b_1 = c_1 + c_2$, and $b_i = c_{i+1}$ for $2 \leqq i \leqq k$. We have $b_1 f(2) = c_2 f(2) + c_1 f(2) \geqq c_2 f(2) + c_1 f(1)$, since $c_1 \geqq 0$ and $f$ is a positive increasing function. Hence, from the conditions of $f$ and $(c_1, c_2, \cdots, c_{k+1})$, we derive two sets of conditions with smaller sizes. The first set is

$$b_1 + b_2 + \cdots + b_k = 0, \quad \text{and} \quad \sum_{i=1}^{j} b_i g(i) \geqq 0 \quad \text{for each } 1 \leqq j \leqq k,$$

where $g(i) = f(i + 1)$ for $1 \leqq i \leqq k$. The second set is

$$c_1 + c_2 = b_1, \quad c_1 f(1) \geqq 0, \quad \text{and} \quad c_2 f(2) + c_1 f(1) \geqq 0.$$

Since $g$ has the same properties as $f$, applying the inductive hypothesis to the first set, we have $b_i = 0$ for $1 \leqq i \leqq k$. Thus, $c_i = 0$ for $3 \leqq i \leqq k + 1$. Since $b_1 = 0$, applying the inductive hypothesis to the second set, we have $c_1 = 0$ and $c_2 = 0$. □

We now define the following decision problem.

*Problem* P1. Given $m$, $\omega$, and a Parallel Task System PTS $= (TS, \tau, G)$, is there a preemptive schedule of PTS on $m$ identical processors such that the schedule length is no larger than $\omega$?

THEOREM 4. *Problem* P1 *is strongly* NP-*complete for arbitrary m and empty precedence constraints.*

*Proof.* We reduce the 3-PARTITION problem to Problem P1. Given an instance $A = (a_1, a_2, \cdots, a_{3z})$ of the 3-PARTITION problem, we construct an instance of Problem P1 as follows. Let $m = 4z - 1$ and $\omega = zB(m - 3(z-1)/2)$. Let $TS = U \cup V$, where $U = \{U_1, U_2, \cdots, U_{3z}\}$ and $V = \{V_1, V_2, \cdots, V_z\}$. For each $1 \leqq i \leqq 3z$,

$$\tau(U_i, j) = \begin{cases} \omega + 1 & \text{if } 1 \leqq j < z \\ a_i(m - j + 1) & \text{if } z \leqq j \leqq m. \end{cases}$$

For each $1 \leqq i \leqq z$,

$$\tau(V_i, j) = \begin{cases} \omega + 1 & \text{if } 1 \leqq j < 2z + i - 1 \\ B(m - (2z - i) + 1) & \text{if } 2z + i - 1 \leqq j \leqq m. \end{cases}$$

First, if the 3-PARTITION problem has a solution, i.e., there exist index sets $I_1, I_2, \cdots, I_z$ such that $\sum_{i \in I_k} a_i = B$ for $1 \leqq k \leqq z$, then we can construct a schedule, as shown in Fig. 5, by executing $V_k$ in parallel on $2z + k - 1$ processors while executing $U_i$ for all $i \in I_k$ in parallel on $2z - k$ processors, for each $1 \leqq k \leqq z$. Since $\sum_{i \in I_k} \tau(U_i, 2z - k) = \tau(V_k, 2z + k - 1)$ for $1 \leqq k \leqq z$ and $\sum_{k=1}^{z} \sum_{i \in I_k} \tau(U_i, 2z - k) = \sum_{k=1}^{z} \tau(V_k, 2z + k - 1) = \omega$, the schedule is a solution to Problem P1.

Next, we show that if there is a schedule $S$ of $TS$ on $m$ processors with length $\omega$ or less, then the 3-PARTITION problem must have a solution. From our construction, it is easy to see that every task in $U$ must execute in parallel on at least $z$ processors and that it is more advantageous to use as many processors as possible. However, every $V_i$, $1 \leqq i \leqq z$, must execute in parallel on at least $2z + i - 1$ processors, but there is no advantage to use more. Thus, we may assume that in $S$, $V_i$ executes in parallel on exactly $2z + i - 1$ processors. Furthermore, since the tasks in $V$ all execute in parallel on $2z$ or more processors, no two of them can execute simultaneously in $S$. Without loss of generality, we may assume that $V_i$ executes in parallel on processors $P_1, P_2, \cdots, P_{2z+i-1}$

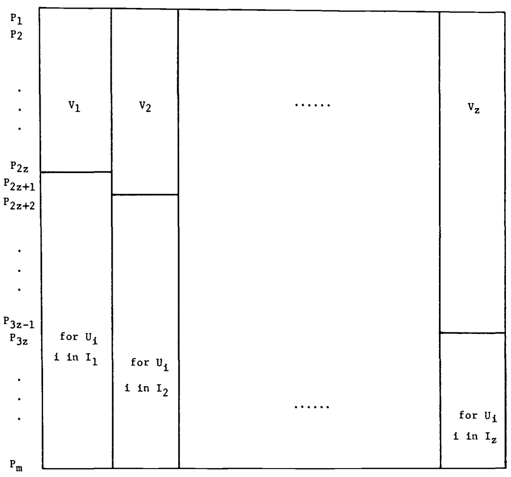


FIG. 5. *Optimal schedule in Theorem 4.*

in $S$. Since $\sum_{i=1}^{z} \tau(V_i, 2z+i-1) = \omega$, the tasks in $U$ must all execute in the remaining processors simultaneously with the tasks in $V$.

Let $I_k$, $1 \leqq k \leqq z$, be a subset of $\{1, 2, \cdots, 3z\}$ such that for $i \in I_k$, task $U_i$ executes in parallel on $2z - k$ processors in $S$. We have $I_1 \cup I_2 \cup \cdots \cup I_z = \{1, 2, \cdots, 3z\}$. Since no two tasks in $U$ can execute simultaneously in $S$, the following conditions must be satisfied:

$$\sum_{k=1}^{l} \sum_{i \in I_k} \tau(U_i, 2z-k) \leqq \sum_{k=1}^{l} \tau(V_k, 2z+k-1) \quad \text{for each } 1 \leqq l \leqq z.$$

This inequality can be rewritten as

$$(*) \qquad \sum_{k=1}^{l} \left(B - \sum_{i \in I_k} a_i\right)(m-(2z-k)+1) \geqq 0 \quad \text{for each } 1 \leqq l \leqq z.$$

It is observed that

$$(**) \qquad \sum_{k=1}^{z} \left(B - \sum_{i \in I_k} a_i\right) = 0.$$

For each $1 \leqq k \leqq z$, we let $c_k = B - \sum_{i \in I_k} a_i$ and $f(k) = m - (2z-k) + 1$. Since $f$ is a positive increasing function, by $(*)$, $(**)$, and Lemma 2, we have $c_k = 0$ for each $1 \leqq k \leqq z$. Thus, $I_1, I_2, \cdots, I_z$ constitute a solution of the 3-PARTITION problem. □

THEOREM 5. *Problem* P1 *is* NP-*complete for* $m = 2$ *and empty precedence constraints.*

*Proof.* We show a reduction of the PARTITION problem to Problem P1. Let $A = (a_1, a_2, \cdots, a_z)$ be an instance of the PARTITION problem. We construct an instance of Problem P1 as follows. Let $\omega = 3B/4 - \frac{1}{2}$ and let $TS = \{T_1, T_2, \cdots, T_{z+1}\}$. $\tau$ is defined as follows.

$$\tau(T_i, 1) = a_i \text{ and } \tau(T_i, 2) = a_i(\tfrac{1}{2} - 1/B), \quad \text{for each } 1 \leqq i \leqq z,$$

and

$$\tau(T_{z+1}, 1) = B/2 \quad \text{and} \quad \tau(T_{z+1}, 2) = B/4 + 1.$$

First, if the PARTITION problem has a solution, say for index sets $I_1$ and $I_2$ such that $I_1 \cup I_2 = \{1, 2, \cdots, z\}$ and $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i = B/2$, then we can construct a schedule as follows. Execute $T_{z+1}$ on processor $P_1$ from time 0 to $B/2$. At the same time, execute $T_i$, $i$ in $I_1$, on processor $P_2$. When this is done, execute the remaining tasks (the ones with indices in $I_2$) in parallel on two processors. Since $\sum_{i \in I_1} a_i = B/2$, the length of the schedule is $B/2 + \sum_{i \in I_2} a_i(\frac{1}{2} - 1/B) = \omega$.

Next, we prove that if the PARTITION problem has no solution, then there can be no solution to Problem P1. It is easy to see that if task $T_{z+1}$ is a 2-processor task with respect to a schedule, then the length of that schedule is at least $3B/4$, which is larger than $\omega$. So in the following, we assume $T_{z+1}$ is a 1-processor task in any schedule to be mentioned. Suppose that $S$ is an optimal preemptive schedule of $TS$ on 2-processors. Let $I_1$ be the subset of $\{1, 2, \cdots, z\}$ such that each task $T_i$, $i$ in $I_1$, is a 1-processor task with respect to $S$. Let $I_2 = \{1, 2, \cdots, z\} - I_1$. Since there is no partition of $A$, either $\sum_{i \in I_1} a_i < B/2$ or $\sum_{i \in I_1} a_i > B/2$. If $\sum_{i \in I_1} a_i < B/2$, then $\sum_{i \in I_2} a_i > B/2$. Since $T_{z+1}$ is a 1-processor task with respect to $S$, the length of $S$ is $B/2 + \sum_{i \in I_2} a_i(\frac{1}{2} - 1/B) > \omega$. On the other hand, if $\sum_{i \in I_1} a_i > B/2$, then $\sum_{i \in I_2} a_i < B/2$, and hence the schedule length is at least $B/2 + (\sum_{i \in I_1} a_i - B/2)/2 + \sum_{i \in I_2} a_i(\frac{1}{2} - 1/B) > \omega$. Hence, the length of $S$

is larger than $\omega$ in both cases. Since $S$ is optimal, there can be no solution to Problem P1. $\square$

We now give a pseudo-polynomial time algorithm to find an optimal preemptive schedule for a Parallel Task System PTS = $(TS, \tau, G)$ with empty precedence constraints for each $m \geqq 2$. The basic idea of our algorithm is as follows. For each schedule $S$ of the PTS, there corresponds a Multiprocessor Task System MTS = $(TS, \tau, G)$ in which task $T_i$ is a $k$-processor task in MTS if $T_i$ is a $k$-processor task with respect to $S$. By the result in [1], finding an optimal preemptive schedule for a Multiprocessor Task System can be done in polynomial time for each $m \geqq 2$. Thus, all we need to do is generate all possible MTSs that correspond to some schedule of the PTS, find an optimal preemptive schedule for each of the generated MTSs, and choose the shortest schedule among all. The bulk of our work is to show that the number of MTSs that we need to generate is bounded above by a pseudo-polynomial function of the size of the PTS.

We begin by presenting a result on a Multiprocessor Task System with empty precedence constraints. Let $S$ be a schedule of a Multiprocessor Task System MTS = $(TS, \tau, G)$, where $TS = \{T_1^1, T_2^1, \cdots, T_{n1}^1\} \cup \{T_1^2, T_2^2, \cdots, T_{n_2}^2\} \cup \cdots \cup \{T_1^m, T_2^m, \cdots, T_{n_m}^m\}$, and $T_i^k$, $1 \leqq i \leqq n_k$, is a $k$-processor task. We divide $S$ into $N$ segments, $S_1, S_2, \cdots, S_N$, where the boundary of a segment is when task assignment on the processors changes. The length of segment $S_i$, $1 \leqq i \leqq N$, is denoted by $l_i$. We let $L = (l_1, l_2, \cdots, l_N)$. For each $S_i$, we define a multiset $K_i = \{k_{i,1}, k_{i,2}, \cdots, k_{i,N_1}\}$, where each $k_{i,j}$, $1 \leqq j \leqq N_i$, corresponds to a $k_{i,j}$-processor task executing in $S_i$. Note that $\sum_{j=1}^{N_1} k_{i,j} \leqq m$ for each $1 \leqq i \leqq N$. We let $K = (K_1, K_2, \cdots, K_N)$. The pair $(K, L)$ is called a preemptive schedule scheme induced by the schedule $S$. Shown in Fig. 6 is a schedule $S$ of an MTS, where $S$ is divided into five segments. We have $L = (5, 3, 6, 3, 2)$ and $K = (\{1, 2, 3\}, \{1, 2, 2, 1\}, \{3, 3\}, \{2, 2, 2\}, \{1, 1, 1\})$. In general, any pair $(K, L)$, where $L$ is a list of $N$ real numbers and $K$ is a list of $N$ multisets of integers with each multiset totaling no more than $m$, is called a preemptive schedule scheme. Notice that every schedule $S$ of an MTS induces a preemptive schedule scheme $(K, L)$. However, it is not the case that every preemptive schedule scheme corresponds to a schedule of the MTS. A preemptive schedule scheme $(K, L)$ is said to be feasible for an MTS if it corresponds to a schedule of the MTS.

$MTS = (TS, \tau, G)$

$TS = \{T_1^1, T_2^1, T_3^1, T_4^1\} \cup \{T_1^2, T_2^2, T_3^2\} \cup \{T_1^3, T_2^3, T_3^3\}$

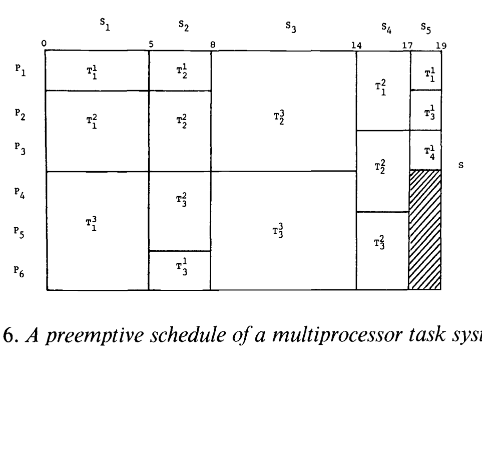
$\tau(T_1^1) = 7,\ \tau(T_2^1) = 2,\ \tau(T_3^1) = 5,\ \tau(T_4^1) = 2$

$\tau(T_1^2) = 8,\ \tau(T_2^2) = 6,\ \tau(T_3^2) = 6$

$\tau(T_1^3) = 5,\ \tau(T_2^3) = 6,\ \tau(T_3^3) = 6$

$K_1 = \{1,2,3\}$ $K_2 = \{1,2,2,1\}$ $K_3 = \{3,3\}$ $K_4 = \{2,2,2\}$ $K_5 = \{1,1,1\}$

$l_1 = 5$ $l_2 = 3$ $l_3 = 6$ $l_4 = 3$ $l_5 = 2$



$S_1$ $S_2$ $S_3$ $S_4$ $S_5$
0 5 8 14 17 19
$P_1$ $P_2$ $P_3$ $P_4$ $P_5$ $P_6$
$T_1^1$ $T_2^1$ $T_1^1$
$T_1^2$ $T_2^2$ $T_2^3$ $T_1^2$ $T_3^1$
$T_4^1$ $T_2^2$
$T_3^2$ $T_1^3$ $T_3^3$ $T_3^2$
$T_3^1$
S

FIG. 6. *A preemptive schedule of a multiprocessor task system.*

In the following, we shall establish necessary and sufficient conditions for a preemptive schedule scheme $(K, L)$ to be feasible for MTS $= (TS, \tau, G)$. For the rest of the discussions, we shall assume that an MTS satisfies the following two conditions. First, the number of $k$-processor tasks in the MTS is at least $\lfloor m/k \rfloor$, for each $1 \leqq k \leqq m$. For otherwise, we can always add in zero-execution-time tasks to make it so. Secondly, the tasks have been indexed in nonincreasing order of execution time. Thus, we have $\tau(T_1^k) \geqq \tau(T_2^k) \geqq \cdots \geqq \tau(T_{n_k}^k)$ and $n_k \geqq \lfloor m/k \rfloor$, for each $1 \leqq k \leqq m$. Let $(K, L)$ be a preemptive schedule scheme, where $K = (K_1, K_2, \cdots, K_N)$ and $L = (l_1, l_2, \cdots, l_N)$. For each $1 \leqq k \leqq m$ and $1 \leqq j \leqq \lfloor m/k \rfloor$, we define $E_j^k = \{ i \mid 1 \leqq i \leqq N$ and there are at least $j$ integers in $K_i$ equal to $k \}$ and

$$D_j^k = \sum_{i \in E_j^k} l_i.$$

Note that $E_{j+1}^k \subseteq E_j^k$ and $D_{j+1}^k \leqq D_j^k$. For each $1 \leqq k \leqq m$, we define a processor system $P^k = \{P_1^k, P_2^k, \cdots, P_{\lfloor m/k \rfloor}^k\}$, where the available processing time on processor $P_j^k$ is from time 0 to time $D_j^k$. Figure 7 shows the processor systems defined by the preemptive schedule scheme of Fig. 6. We are now ready to establish necessary and sufficient conditions for a preemptive schedule scheme to be feasible for a Parallel Task System.

LEMMA 3. *Let $(K, L)$ be a preemptive schedule scheme and let $D_j^k$ be defined as above. $(K, L)$ is feasible for an* MTS *if and only if the following conditions are satisfied for all* $1 \leqq k \leqq m$ *and* $1 \leqq j \leqq \lfloor m/k \rfloor$:

$$\sum_{i=1}^{j} D_i^k \geqq \begin{cases} \sum_{i=1}^{j} \tau(T_i^k) & \text{if } 1 \leqq j < \lfloor m/k \rfloor \\ \sum_{i=1}^{n_k} \tau(T_i^k) & \text{if } j = \lfloor m/k \rfloor. \end{cases}$$

*Proof.* Noting that $D_j^k$, $1 \leqq j \leqq \lfloor m/k \rfloor$, define a processor system for the $k$-processor tasks, the lemma can be proved by the same technique as in [4], [5]. Because of space limitation, we shall omit the proof here. □

LEMMA 4. *Let* MTS$_1$ *and* MTS$_2$ *be two Multiprocessor Task Systems. Then, they have the same optimal preemptive schedule lengths if for each* $1 \leqq k \leqq m$, *the execution times of the largest* $(\lfloor m/k \rfloor - 1)$ *k-processor tasks in* MTS$_1$ *are the same as those in*

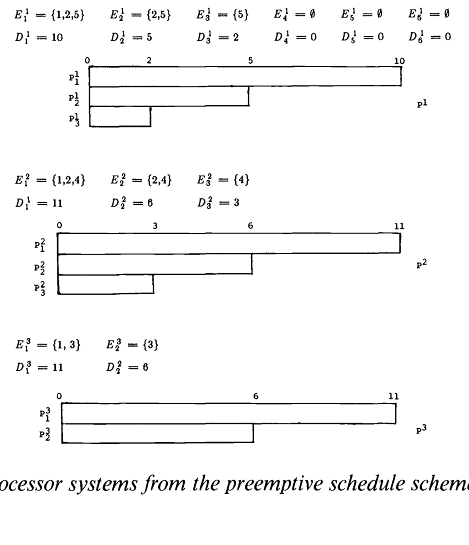


FIG. 7. *Processor systems from the preemptive schedule scheme of Fig.* 6.

MTS$_2$, *and the total execution time of the remaining k-processor tasks are the same in both task systems.*

*Proof.* Let $S_1$ be an arbitrary preemptive schedule for MTS$_1$ and let $(K_1, L_1)$ be the preemptive schedule scheme induced by $S_1$. We have that $(K_1, L_1)$ is feasible for MTS$_1$. Since for each $1 \leqq k \leqq m$, the execution times of the largest $(\lfloor m/k \rfloor - 1)$ $k$-processor tasks in MTS$_2$ are the same as those in MTS$_1$ and the total execution time of the remaining $k$-processor tasks in MTS$_2$ is the same as that in MTS$_1$, by Lemma 3, $(K_1, L_1)$ is also feasible for MTS$_2$. Thus, there is a preemptive schedule $S'_1$ for MTS$_2$ with the same length as $S_1$. Similarly, if $S_2$ is an arbitrary preemptive schedule for MTS$_2$, then there is a preemptive schedule $S'_2$ for MTS$_1$ with the same length as $S_2$. Therefore, the optimal preemptive schedule lengths for both task systems must be identical. □

From Lemma 4, we see that the optimal preemptive schedule length of an MTS is completely determined by $H = \sum_{k=1}^{m} \lfloor m/k \rfloor$ integers, namely, the execution times of the largest $(m-1)$ 1-processor tasks, the total execution time of the remaining 1-processor tasks, the execution times of the largest $(\lfloor m/2 \rfloor - 1)$ 2-processor tasks, the total execution time of the remaining 2-processor tasks, $\cdots$, and the total execution time of all $m$-processor tasks. For every preemptive schedule $S$ of a PTS, there corresponds an MTS in which a task is a $k$-processor task if it is a $k$-processor task with respect to $S$. By Lemma 4, the number of MTSs we need to examine is bounded above by $M^H$, where $M = \max_{k=1}^{m} \{\sum_{i=1}^{n_k} \tau(T_i^k)\}$. Since $H$ is $O(m \log m)$ and since finding an optimal preemptive schedule for an MTS can be done in polynomial time, the running time of the entire algorithm is bounded above by a pseudo-polynomial function of the size of the input.

Before we finish this section, we briefly describe how one can generate all the MTSs that need to be examined for any given PTS. Let PTS $= (TS, \tau, G)$, where $TS = \{T_1, T_2, \cdots, T_n\}$. We generate a $H$-dimensional table $F(i, x_1, x_2, \cdots, x_{H-1})$, $0 \leqq i \leqq n$, $0 \leqq x_1, x_2, \cdots, x_{H-1} \leqq M$, where $F(i, x_1, x_2, \cdots, x_{H-1})$ gives the smallest total execution time of the $m$-processor tasks, among all schedules of the first $i$ tasks of $TS$, such that the largest execution time of the 1-processor tasks in the schedule is $x_1$, the second largest execution time of the 1-processor tasks in the schedule is $x_2$, $\cdots$, and the total execution time of the $(m-1)$-processor tasks in the schedule is $x_{H-1}$. Note that $F(i, x_1, x_2, \cdots, x_{H-1}) = \infty$ if there is no such schedule. The MTSs that need to be examined are those where $F(n, x_1, x_2, \cdots, x_{H-1}) < \infty$. It is readily verified that the table $F$ can be constructed in $O(nHM^H)$ time. As an example, let us consider $m = 3$. In this case, $H = 5$. $F(0,0,0,0,0) = 0$ and $F(0, x_1, x_2, x_3, x_4) = \infty$ for $0 < x_1, \cdots, x_4 \leqq M$. For each $1 \leqq i \leqq n$ and $0 \leqq x_1, \cdots, x_4 \leqq M$, we compute $F(i, x_1, x_2, x_3, x_4)$ as follows: First, compute

$$X = \begin{cases} F(i-1, x_1, x_2, x_3 - \tau(T_i, 1), x_4) & \text{if } x_1 \geqq x_2 > \tau(T_i, 1) \\ \min_{0 \leqq y \leqq x_2}\{F(i-1, x_2, y, x_3 - y, x_4)\} & \text{if } \tau(T_i, 1) = x_1 > x_2 \\ \min_{0 \leqq y \leqq x_2}\{F(i-1, x_1, y, x_3 - y, x_4)\} & \text{if } x_1 \geqq x_2 = \tau(T_i, 1) \\ \infty & \text{otherwise.} \end{cases}$$

Now,

$$F(i, x_1, x_2, x_3, x_4) = \min\{F(i-1, x_1, x_2, x_3, x_4) + \tau(T_i, 3),$$
$$F(i-1, x_1, x_2, x_3, x_4 - \tau(T_i, 2)), X\}.$$

From the above discussions, we have the following theorem.

THEOREM 6. *The problem of finding an optimal preemptive schedule for a* PTS *with empty precedence constraints is solvable in pseudo-polynomial time for each $m \geqq 2$.*

**4. Conclusion.** In this paper, we propose a model of task systems, the so-called Parallel Task Systems, to model tasks that implement parallel algorithms. We study the complexity of finding a minimum length schedule for both nonpreemptive and preemptive scheduling disciplines. The following problems are left open in this paper: (1) For $m = 4$ and empty precedence constraints, is the problem of finding an optimal non-preemptive schedule for a PTS strongly NP-hard? (2) For $m = 2$ and nonempty precedence constraints, is the problem of finding an optimal preemptive schedule for a PTS strongly NP-hard? (3) For arbitrary $m$ and empty precedence constraints, is the problem of finding an optimal preemptive schedule for an MTS strongly NP-hard?

## REFERENCES

[1] J. BLAZEWICZ, M. DRABOWSKI, AND J. WELGARZ, *Scheduling multiprocessor tasks to minimize schedule length*, IEEE Trans. Comput., Vol. C-35, No. 5, 1986, pp. 389–393.

[2] E. G. COFFMAN, JR., ED., *Computer and Job-Shop Scheduling Theory*, John Wiley, New York, 1976.

[3] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.

[4] T. GONZALES AND S. SAHNI, *Preemptive scheduling of uniform processor systems*, Journal of Assoc. Comput. Mach., 25 (1978), pp. 92–101.

[5] E. C. HORVATH, S. LAM, AND R. SETHI, *A level algorithm for preemptive scheduling*, Journal of Assoc. Comput. Mach., 24 (1977), pp. 32–43.

[6] R. MCNAUGHTON, *Scheduling with deadlines and loss functions*, Management Sci., 6 (1959), pp. 1–12.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 488–490, November 1989

# ON TWO CLASSICAL RAMSEY NUMBERS OF THE FORM $R(3, n)$*

GEOFFREY EXOO†

**Abstract.** New lower bounds are given for the classical Ramsey numbers $R(3, 10)$ and $R(3, 12)$. Both constructions were made using a variant of the Metropolis Algorithm and were built on smaller cyclic constructions.

**Key words.** Ramsey numbers, heuristics

**AMS(MOS) subject classification.** 05C55

Two new lower bounds for Ramsey numbers of the form $R(3, n)$ are proved below. Both proofs are completed by extending cyclic constructions for $R(3, n-1)$.

In general our notation follows that of Harary [2]. We use $R(s, t)$ to denote the classical Ramsey number of $K_s$ versus $K_t$, defined to be the smallest integer $n$ such that in any 2-coloring of the edges of $K_n$ there is a monochromatic copy of $K_s$ in color 1 or a monochromatic copy of $K_t$ in color 2. A coloring of a complete graph is called an $(s, t)$-coloring if there are no monochromatic copies of $K_s$ in color 1 or of $K_t$ in color 2. A graph of order $n$ is called a *cyclic* $n(a_1, \cdots, a_k)$ graph if its vertices can be labeled with the integers from 0 to $n-1$ so that two vertices are adjacent if and only if their difference is $a_i$, for some $i$, $1 \leqq i \leqq k$.

The underlying algorithm we used to make these constructions is a procedure that has been called *simulated annealing* [3], and is based on an algorithm devised by Metropolis et al. [4] for application to statistical mechanics. We offer a brief description. Let $f$ be an integer-valued function of integer variables $x_1, \cdots, x_n$, and suppose we wish to find the minimum value of $f$. At each step of the algorithm we have a *current vector* $(x_1, \cdots, x_n)$ that may initially be chosen at random. We consider a small random change in one of the variables $x_i$, yielding a new vector $(x_1, \cdots, x_i', \cdots, x_n)$. The values $y = f(x_1, \cdots, x_i, \cdots, x_n)$ and $y' = f(x_1, \cdots, x_i', \cdots, x_n)$ are compared, and $\Delta Y = y' - y$ is computed. If $\Delta Y \leqq 0$, then the new vector is accepted as the current vector, otherwise the new vector is accepted with probability $\exp(-\Delta Y / k_B T)$, where $k_B$ is the analogue of the Boltzmann constant and $T$ is an analogue of temperature. In the course of running the algorithm we usually begin with a relatively large value for $T$ (i.e., a high temperature), and gradually lower the value of $T$ (i.e., allow the system to cool).

The first problem that arises when applying this procedure to Ramsey numbers is that of determining $f$. This issue has been discussed in some detail in [1]. New issues arise when dealing with far off-diagonal cases. Specifically, with $R(3, t)$, we must decide how much weight to give to a $K_3$ in color 1, as opposed to a $K_t$ in color 2. In the context of the Metropolis Algorithm, the random change in the current vector corresponds to recoloring one edge in a given 2-coloring. Suppose that coloring a given edge in color 1 yields $m_1$ monochromatic $K_3$'s in color 1, while coloring it with color 2 yields $m_2$ monochromatic $K_t$'s in color 2. Let $\rho = m_2/m_1$. The question that must be answered is: For what values of $\rho$ do we prefer color 1 and for what values do we prefer color 2? Let $\rho_0$

* Received by the editors September 28, 1988; accepted for publication February 1, 1989. This research was supported by a grant from the Office of Naval Research.

† Department of Mathematics and Computer Science, Indiana State University, Terre Haute, Indiana 47809.

be the value of $\rho$ for which we are indifferent. In other words, for values of $\rho > \rho_0$ we choose color 1, for values of $\rho < \rho_0$ we choose color 2, and when $\rho = \rho_0$ we make a random choice. In practice we have found that choosing

$$\rho_0 = \binom{t}{2} \Big/ 3$$

is a good choice, so that the weights are inversely proportional to the number of edges in the graphs we are trying to avoid.

The table of Ramsey numbers given in [5] seems to be the most recently published. The values listed there for numbers of the form $R(3, n)$ are as follows:

$$R(3,3)=6, \qquad R(3,4)=9,$$
$$R(3,5)=14, \qquad R(3,6)=18,$$
$$R(3,7)=23, \qquad 28 \leqq R(3,8) \leqq 29,$$
$$R(3,9)=35, \qquad 39 \leqq R(3,10) \leqq 44,$$
$$46 \leqq R(3,11) \leqq 54, \qquad 49 \leqq R(3,12) \leqq 63.$$

We improve the lower bounds for $R(3, 10)$ and $R(3, 12)$ by one.

THEOREM 1. $R(3, 10) \geqq 40$.

*Proof.* Begin with the cyclic $(3, 9)$-coloring of $K_{35}$ given by having the edges of the cyclic graph 35(1, 7, 11, 16, 19, 24, 28, 34) colored in color 1, and the edge of the complement colored in color 2. To this graph we add four vertices labeled $a$, $b$, $c$, and $d$. The edges joining $a$ to $c$ and $b$ to $d$ are colored in color 2. The remaining edges among these four vertices are colored in color 1. In addition, the four new vertices are joined in color 1 to those of the original 35 as listed below:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $a$: | 2 | 15 | 19 | 27 | 32 | | |
| $b$: | 11 | 17 | 25 | 29 | | | |
| $c$: | 8 | 16 | 26 | 28 | 34 | | |
| $d$: | 1 | 4 | 10 | 18 | 22 | 24 | 30. |

The remaining edges are in color 2.

THEOREM 2. $R(3, 12) \geqq 50$.

*Proof.* The construction proceeds just as in Theorem 1. We begin with the cyclic coloring derived from the graph 45(3, 10, 11, 12, 16, 29, 33, 34, 35, 42). Again we add four vertices, $a$, $b$, $c$, and $d$, with $a$ adjacent to $c$ in color 2 and $b$ adjacent to $d$ in color 2. All other edges among these four vertices are in color 1. The color 1 edges joining the new vertices to the original 45 are given below:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $a$: | 1 | 22 | 24 | 31 | 39 | | | |
| $b$: | 6 | 7 | 14 | 28 | 33 | 37 | | |
| $c$: | 10 | 12 | 27 | 34 | 35 | 36 | 40 | |
| $d$: | 2 | 3 | 4 | 17 | 21 | 25 | 30 | 43. |

We note that evidence seems to be accumulating for the conjecture that it is the exception, rather than the rule, for Ramsey colorings to be cyclic.

## REFERENCES

[1] G. Exoo, *Constructing Ramsey graphs with a computer*, Congr. Numer., 59 (1987), pp. 31–36.

[2] F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.

[3] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

[4] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Equation of state calculations for fast computing machines*, J. Chem. Phys., 6 (1953), p. 1087.

[5] S. P. Radziszowski and D. L. Kreher, *Search algorithm for Ramsey graphs by union of group orbits*, J. Graph Theory, 12 (1988), pp. 59–72.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 491–499, November 1989

# PAIR LABELLINGS WITH GIVEN DISTANCE*

Z. FUREDI†, J. R. GRIGGS‡, AND D. J. KLEITMAN§

**Abstract.** Given a graph $G$ and $d \in \mathbb{Z}^+$, the pair labelling number, $r(G, d)$, is defined to be the minimum $n$ such that each vertex in $G$ can be assigned a pair of numbers from $\{1, \cdots, n\}$ in such a way that any two numbers used at adjacent vertices differ by at least $d$. A question of Roberts' is answered by determining all possible values of $r(G, d)$ given the chromatic number of $G$. The answer follows by determining the chromatic number of the graph that has pairs of integers as vertices and edges joining pairs that are distance at least $d$ apart. For general $t \in \mathbb{Z}^+$, the analogous questions for $t$-sets instead of pairs are considered. A solution for general $t$ is conjectured which, for $d = 1$, reduces to Lovász's theorem on Kneser graphs.

**Key words.** generalized graph colorings, Kneser graphs

**AMS(MOS) subject classification.** 05C15

**1. Introduction.** There has been a considerable effort [CR], [R1] to study properties of "$T$-colorings" of graphs in which a set of nonnegative integers $T$ is specified, and each vertex of a simple graph $G = (V, E)$ is assigned a "color," denoted $f(v)$, where $f(v)$ is a positive integer and for every edge $\{v, w\} \in E$ the value $|f(v) - f(w)| \notin T$. Roberts [R2] has proposed an analogous problem in which each vertex $v$ is assigned an unordered *pair* of integers as its color subject to the restrictions that adjacent vertices never receive the same or adjacent integers. The proposed problem is motivated by the task of assigning channel frequencies without interference. Our investigations here will find a close connection between this theory and Kneser graphs.

Throughout the paper, sets denoted by interval notation, such as $[1, n]$, are restricted to integer values. For a set $S$ and value $t \in \mathbb{Z}^+$, $\binom{S}{t}$ denotes the collection of all (unordered) $t$-subsets of $S$. All graphs $G = (V, E)$ are simple and undirected, i.e., $E \subseteq \binom{V}{2}$.

A *pair labelling* of a graph $G = (V, E)$ is a function $f : V \to \binom{\mathbb{Z}^+}{2}$. We are interested in pair labellings such that no vertex receives a label that is too close to that of a neighbor. The *distance* between two pairs $A, B \subseteq \binom{\mathbb{Z}^+}{2}$ is defined to be the minimum value of $|a - b|$ over all $a \in A$ and $b \in B$. A pair labelling $f$ of a graph $G$ has *distance* $d(f)$ where $d(f)$ is the minimum, over all edges $\{v, w\} \in E$, of the distance between the pairs $f(v)$ and $f(w)$. We wish to study, for given graph $G$ and distance $d$, the minimum number $n$ such that there exists a pair labelling $f$ with distance $d(f) \geqq d$ and $\max_v f(v) = n$. That is, we seek to minimize $n$ such that there exists $f : V \to \binom{[1,n]}{2}$ with $d(f) \geqq d$. Let $r(G, d)$ denote this minimum.

More generally, for any $t \in \mathbb{Z}^+$, a *t-labelling* of a graph $G$ is a function $f : V \to \binom{\mathbb{Z}^+}{t}$. We may extend the definitions above to distance between $t$-sets and distance $d(f)$ of a $t$-labelling $f$. Let $r_t(G, d)$ denote the minimum $n$ such that there exists $f : V \to \binom{[1,n]}{t}$ with $d(f) \geqq d$. Notice that the special case $t = 1$ and $d = 1$ is the familiar one of

* Received by the editors May 9, 1988; accepted for publication (in revised form) March 21, 1989. This research was done while the authors were visiting the Institute for Mathematics and its Applications at the University of Minnesota, Minneapolis, Minnesota 55455.

† Mathematical Institute of the Hungarian Academy of Sciences, 1364 Budapest, P.O. 127, Hungary. This research was supported in part by Hungarian National Foundation for Scientific Research grant 1812.

‡ Mathematics Department, University of South Carolina, Columbia, South Carolina 29208. This research was supported in part by National Science Foundation grant DMS 87-01475.

§ Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. This research was supported in part by National Science Foundation grant DMS 86-06225 and Air Force Office of Scientific Research grant OSR-86-0076.

vertex-coloring for graphs, so that $r_1(G, 1) = \chi(G)$, the chromatic number of $G$. Of course, $r_2(G, d)$ is the same as $r(G, d)$.

The pair labelling number $r(G, d)$ is a special case of what Roberts calls the *T-span* of $G$, denoted $sp_T(G)$. Given a set $T$ of nonnegative integers, $sp_T(G)$ is the smallest integer $n$ such that there exists a pair labelling $f$ of $G$ using integers in $[1, n]$ with the property that whenever $\{v, w\}$ is an edge in $G$, $a \in f(v)$, and $b \in f(w)$, then $|a - b| \notin T$. Thus $r(G, d)$ is $sp_T(G)$ where $T = \{0, 1, \cdots, d - 1\}$. In general the $t$-labelling number $r_t(G, d)$ is similarly the minimum span, denoted $sp_T^n(G)$ by Roberts, of an $n$-tuple $T$-coloring of $G$ where $T = \{0, 1, \cdots, d - 1\}$. An up-to-date survey of this sort of generalized colorings has been prepared by Roberts [R3].

How can we efficiently label the complete graph $K_k$? Given $t$ and $d$, we can assign the first $t$ integers to some vertex, skip the next $d - 1$ integers, assign the next $t$ integers to a second vertex, skip the next $d - 1$ integers, and so on. It is easily checked that no other labelling of a complete graph is as efficient. Hence $r_t(K_k, d) = kt + (k - 1)(d - 1)$, and the given labelling is the only one that attains $r_t(K_k, d)$ up to permuting the vertices. The same labelling strategy works more generally for any $k$-chromatic graph: Given a $k$-coloring of $V$, we can replace the first color by the first $t$ integers, then skip $d - 1$ integers, and replace the second color by the next $t$ integers, etc. We have proved the following result.

PROPOSITION 1.1. *Let $t, d, k \in \mathbb{Z}^+$. Suppose $G$ is a graph with $\chi(G) = k$. Then $r_t(G, d) \leqq kt + (k - 1)(d - 1)$, and this bound is sharp for $G = K_k$.*

This establishes the close connection between the chromatic number and labelling numbers. In the special case $t = 2$ and $d = 2$ it is the motivation for the following problem posed by Roberts [R2]: Determine the range of possible values of $r(G, 2)$ for arbitrary $k$ where $k = \chi(G)$.

In the next section we explore the connection between $t$-labellings and graph homomorphisms. For labellings we state our main theorem, which answers Roberts' question. More generally, for arbitrary $d$ we determine the range of possible values of $r(G, d)$ for graphs $G$ with given chromatic number. The proof is reduced to one essential lemma that gives a lower bound on the chromatic number of a particular class of graphs. The lemma, which is of independent interest in light of its connection to Kneser graphs, has a purely graph-theoretic proof, given in § 3. Further discussion of pair labellings follows in § 4. The paper concludes with a conjecture about what happens for $t$-labellings for general $t$. A weaker version of the main theorem for pair labellings can be proven for general $t$. A purely graph-theoretic proof of the general conjecture would be surprising, since the conjecture yields the chromatic number of Kneser graphs as a special case.

**2. $t$-labellings, graph homomorphisms, and the chromatic number.** When we study vertex-labellings of graphs, it is often helpful to consider graph homomorphisms. That is the case here. A *graph homomorphism* from a graph $G = (V, E)$ to a graph $H = (W, F)$ is a map $g: V \to W$ that sends edges to edges, i.e., for all $\{v, w\} \in E$, $\{g(v), g(w)\} \in F$. We say $G$ is *homomorphic to* $H$ if there exists such a homomorphism from $G$ to $H$. In this language, a graph $G$ has a $k$-coloring, i.e., $\chi(G) \leqq k$, if and only if $G$ is homomorphic to $K_k$. For related work on homomorphisms, see [A], [G], [HN].

In a similar way we may view $t$-labellings as homomorphisms to the graph of labels. We introduce the *t-graph* $G_t(n, d)$ with vertex set $V = \binom{[1,n]}{t}$ and edge set $E$ that contains every pair $\{A, B\} \in \binom{V}{2}$ such that the distance between $A$ and $B$ is at least $d$. If $t = 2$ we suppress the $t$ and write $G(n, d)$, which we call the *pair graph*. It follows from the definitions for any $G, t, d, n$ that $r_t(G, d) \leqq n$ if and only if $G$ is homomorphic to the $t$-graph $G_t(n, d)$. For the case proposed for study by Roberts, $t = 2$ and $d = 2$, we discuss characterizations by homomorphisms in more detail in § 4.

Given this homomorphism characterization of $t$-labellings, it is clear that the chromatic numbers for the $t$-graphs $G_t(n, d)$ are of particular importance in our study. Suppose $t$, $d$, $k$ are given and $n$ is the smallest value such that $\chi(G_t(n, d)) \geqq k$. Then consider any graph $G$ with $r_t(G, d) < n$. $G$ is homomorphic to $G_t(n-1, d)$, which is $(k-1)$-colorable and hence homomorphic to $K_{k-1}$. By composition, $G$ is homomorphic to $K_{k-1}$, i.e., $\chi(G) \leqq k-1$. Thus the range of possible values of $r_t(G, d)$ for graphs $G$ with $\chi(G) = k$ is contained in the interval $[n, kt + (k-1)(d-1)]$.

For pair labellings we can determine the minimum value $n$ above and show that for this $n$, $\chi(G(n, d)) = k$, so that this $n$ is one of the attainable values of $r(G, d)$ for given $k$. Then we prove by induction on $k$ that every value in the interval is attained. We now state our main result which contains the answer to Roberts' question.

THEOREM 2.1. *Let $d, k \in \mathbb{Z}^+$. Suppose the graph $G$ has $\chi(G) = k$. If $k = 1$, then $r(G, d) = 2$. If $k \geqq 2$, then $r(G, d) \in [d(k-1)+3, d(k-1)+k+1]$, and all values in this interval are attained by suitable graphs $G$.*

*Proof.* If $k = 1$ and $\chi(G) = k$, then $G$ consists of one or more isolated vertices. Trivially, $r(G, d) = 2$ for all $d$ in this case. Then suppose $k \geqq 2$ and $\chi(G) = k$. By Proposition 1.1, $r(G, d) \leqq d(k-1)+k+1$, and this bound is sharp. Next we show that $r(G, d) \geqq d(k-1)+3$. In view of the discussion above, this follows by bounding $\chi(G(d(k-1)+2, d))$ above by $k-1$. We now prove a more general result that applies for all $t$. □

PROPOSITION 2.2. *Let $t, d, k \in \mathbb{Z}^+$. Then*

$$\chi(G_t(d(k-1)+2t-2, d)) \leqq k-1.$$

*Proof of Proposition* 2.2. It suffices to describe a suitable $(k-1)$-coloring of the vertices of $G_t(d(k-1)+2t-2, d)$. For $1 \leqq m \leqq k-2$ assign color $m$ to all vertices ($t$-subsets) $\{i_1 < i_2 < \cdots < i_t\}$ such that $i_1 \in [d(m-1)+1, dm]$. The remaining uncolored vertices form the set $\binom{I}{t}$, where $I = [d(k-2)+1, d(k-1)+2t-2]$. Let $A \in \binom{I}{t}$ be any such vertex. Then at least $d+t-1$ elements of $I$ are within distance $d-1$ of some element of $A$. Since $|I| = d+2t-2$, any other vertex $B \in \binom{I}{t}$ contains some element within distance $d-1$ of $A$. Hence every vertex in $\binom{I}{t}$ may be assigned color $k-1$. □

We have shown that the bounds in the theorem are correct and that the upper bound is sharp. Next we prove that the lower bound is also sharp. We must show there is a graph $G$ such that $\chi(G) = k$ and $r(G, d) = d(k-1)+3$. Such a graph $G$ must be homomorphic to $G(d(k-1)+3, d)$ which, by Proposition 2.2, has chromatic number at most $k$. If we can establish that $\chi(G(d(k-1)+3, d)) = k$, then this graph can be the $G$ we seek. This follows from Lemma 2.3.

LEMMA 2.3. *Let $d, k \in \mathbb{Z}^+$. Then $\chi(G(d(k-1)+3, d)) \geqq k$.*

The proof of this lemma is the most demanding part of our paper, and we devote § 3 to it. Assume here that this lemma is true. To complete the proof of the theorem it then remains to produce $k$-chromatic graphs that assume the intermediate values of the pair labelling number $r$. We prove this by induction on $k$. For $k = 1$ and 2 the only possible values for $r(G, d)$ are 2 and $d+3$, respectively, so we know they are realized by suitable $G$ (e.g., $K_1$ and $K_2$, respectively).

Assume for induction that $k \geqq 2$ and every value $i$ in the interval

$$[d(k-1)+3, d(k-1)+k+1]$$

is realized by a suitable graph $G_i = (V_i, E_i)$. Attach a new vertex $w$ that is adjacent to all of $G_i$, i.e., let $G'_i = (V'_i, E'_i)$, where $V'_i = V_i \cup \{w\}$ and $E'_i = E_i \cup \{\{v, w\} : v \in V_i\}$.

It follows immediately that

$$\chi(G_i') = \chi(G_i) + 1 = k + 1,$$

$$\text{and } r(G_i') = r(G_i, d) + d + 1 = i + d + 1.$$

Hence the graphs $G_i'$ all have chromatic number $k + 1$ and exhibit the following values of $r(G, d)$ : $[dk + 4, dk + (k + 1) + 1]$. This includes all values in the interval except $dk + 3$, which is handled by Lemma 2.3. □

**3. Proof of Lemma 2.3.** We interpret the lemma in the following way: Pairs in $[1, n]$ correspond to edges in the complete graph $K_n$. So we must show that for all $d$, $k \geqq 1$ it is impossible to partition the edges of $K_n$ into $k - 1$ graphs $G_i$, i.e., $E(K_n) = E(G_1) \cup \cdots \cup E(G_{k-1})$, when $n = d(k-1) + 3$, such that no $G_i$ contains distinct edges $e$ and $f$ unless $|a - b| < d$ for some $a \in e$, $b \in f$. When edges $e$, $f$ have $|a - b| < d$ for some $a \in e$, $b \in f$, we say that the edges are "close" to one another; otherwise, they are "far." In these terms, we seek to prove that there is no $k - 1$-coloring such that all edges of the same color are close to one another. For $d = 1$ the desired result, i.e., $\chi(G(k + 2, 1)) = k$, is a special case ($t = 2$) of the well-known result about Kneser graphs (cf. § 5). For the sake of completeness we supply a direct proof here. For $d \geqq 2$ we prove the result for a more general class of graphs by induction on $k$.

First suppose $d = 1$ and $n = k + 2$. We use induction on $k$. The result is trivial for $k = 1$. Suppose $k \geqq 2$. Suppose $E(K_n) = E(G_1) \cup \cdots \cup E(G_m)$ where $m \leqq k - 1$ and for all $i$ any two edges in $E(G_i)$ intersect. Then the edges in any color class $E(G_i)$ either form a triangle or a star (i.e., have a common vertex). Since $|E(K_n)| = \binom{k+2}{2} > 3(k - 1) \geqq 3m$, some color class must be a star through some vertex $j$. Then the partition of $E(K_n)$ induces a partition of the edges in the subgraph of $K_n$ on vertices $[1, n]/\{j\}$ into just $m - 1 \leqq k - 2$ color classes, which is impossible by induction on $k$. Therefore $m \geqq k$, and hence $\chi(G(k + 2, 1)) = k$ as claimed.

Let $d \geqq 2$. For our induction on $k$ to succeed we must consider edge-colorings of a larger class of graphs, called *cut-graphs*. Suppose $V = \{i_1, i_2, \cdots, i_a\}$, where $1 \leqq i_1 < i_2 < \cdots < i_a \leqq n$. A *cut* between vertices that are consecutive in $V$, say between $i_j$ and $i_{j+1}$, written $i_j | i_{j+1}$, will mean two things. First, the edge $\{i_j, i_{j+1}\}$ is removed from the graph. Second, vertices below the cut, including $i_j$, are considered far, i.e., distance at least $d$, from vertices above the cut, including $i_{j+1}$.

Consider an example. Let $d = 3$ and $V = \{1, 4, 5, 6, 8, 9\}$. The graph $G = 1, 4|5, 6|8|9$ has six vertices and three cuts. Thus $E(G)$ is $\binom{V}{2}$ except for $\{4, 5\}$, $\{6, 8\}$, and $\{8, 9\}$. Due to the cuts, the edge $\{4, 8\}$ is now far (distance at least $d = 3$) from $\{5, 9\}$. The cuts make the graph easier to color by removing edges, yet harder to color by increasing distances.

We shall complete the proof of the lemma by proving for $d \geqq 2$ this stronger statement.

PROPOSITION 3.1. *Let $d \geqq 2$, $k \geqq 1$, and $n \geqq d(k - 1) + 3$. Let $V \subseteq \mathbb{Z}^+$ with $|V| = n$. Let $G$ be any cut-graph on vertex set $V$. Then any coloring of $E(G)$ with distance $d$ requires at least $k$ colors.*

*Proof.* We assume that $V = [n]$ in order to make it as easy to color $G$ as possible. For example, with $d = 3$ the graph $G' = 1, 2|3, 4|5|6$ is no harder to color than $G = 1, 4|5, 6|8|9$. Indeed, far edges in $G$ may even correspond to close edges in $G'$, e.g., $\{1, 8\}$ and $\{4, 9\}$ in $G$ become $\{1, 5\}$ and $\{2, 6\}$ in $G'$. More precisely, any feasible coloring of $G$ induces a feasible coloring of $G'$.

With $d \geqq 2$ fixed, suppose first that $k = 1$, $n \geqq 3$. To prove one color is required it is enough to show there is an edge in any cut-graph on $[1, 3]$. If there is no cut, $\{1, 2\}$ is an edge. If there is one cut, the two vertices on the same side of the cut determine an edge. If there are two cuts, they are $1|2$ and $2|3$. Then $\{1, 3\}$ is an edge. Hence at least one color is required.

We induct on $k$. Suppose $k \geqq 2$ and that the proposition holds for $k - 1$. Let $n \geqq d(k - 1) + 3$, and let $G$ be any cut-graph on $[1, n]$. Suppose, for contradiction, that $G$ has an edge-coloring with $k - 1$ colors. $E(G) = E(G_1) \cup \cdots \cup E(G_{k-1})$, where any two edges in any $G_i$ are close to one another. We say a set of vertices $S \subseteq [1, n]$ is *heavy for color* $i$, written $S \in H_i$, if putting into $G_i$ every edge in $G$ that meets some vertex in $S$ maintains the property that the distance is at most $d - 1$ between any two edges in $E(G_i)$.

The interval $[1, n]$ is partitioned into subintervals, which we call *sections*, by the cuts in $G$. Suppose some section $S$ has $j$ vertices, say $\{a + 1, \cdots, a + j\}$, where $2 \leqq j \leqq d$. There is some edge $e$ with both ends in $S$, say $e \in E(G_1)$. Since every edge in $E(G_1)$ is distance at most $d - 1$ from $e$, it follows that every edge in $E(G_1)$ meets $S$. We now remove $S$ and all edges that meet $S$ from $G$. If $a \geqq 1$ and $a + j + 1 \leqq n$, then also remove the edge $\{a, a + j + 1\}$ and insert a new cut $a|a + j + 1$. We have a new cut-graph, call it $G'$. All edges in $E(G_1)$ were deleted along with some others possibly, so the color classes $E(G_2) \cup \cdots \cup E(G_{k-1})$ induce a $(k - 2)$-coloring of $E(G')$ with distance $d$. However, the cut-graph $G'$ has $n - j \geqq d(k - 2) + 3$ vertices, and by induction on $k$, it requires at least $k - 1$ colors for $E(G')$, a contradiction. Therefore we may assume hereafter that each section in $G$ has only one vertex or else at least $d + 1$ vertices.

Suppose now that $G$ has single vertex sections on both ends, i.e., $1|2$ and $n - 1|n$. Then $\{1, n\}$ is an edge, say $\{1, n\} \in E(G_1)$. All vertices in $[2, n - 1]$ are distance at least $d$ from 1 and $n$, so every edge in $E(G_1)$ contains 1 or $n$. Thus the induced subgraph $G'$ of $G$ on $[2, n - 1]$ is a cut-graph, and the coloring induced by $E(G)$ uses only $k - 2$ colors. However, $|V(G')| = n - 2 \geqq d(k - 2) + 3$, so by induction at least $k - 1$ colors are required for $G'$, a contradiction.

We assume for the remainder of the proof that at least one end, say the end beginning at 1, has at least $d + 1$ vertices in its section. Suppose there is a cut near the other end, i.e., $n - 1|n$. The edge $\{1, 2\}$ belongs to $G$, say $\{1, 2\} \in E(G_1)$. To be closer than distance $d$ to $\{1, 2\}$, every edge in $E(G_1)$ must meet $[1, d + 1]$. Thus every vertex in $[2, d]$ is distance at most $d - 1$ from every edge in $E(G_1)$. It follows that $[2, d] \in H_1$, so we may recolor every edge meeting $[2, d]$ by color 1 and still have a valid coloring. Assume we have done this.

Since $k \geqq 2$, we have $n \geqq d + 3$, so the edge $\{1, n\}$ is in $E(G)$. If $\{1, n\} \notin E(G_1)$, say $\{1, n\} \in E(G_2)$, then every edge in $E(G_2)$ must contain 1 or $n$. But then the $k - 2$ color classes $E(G_1) \cup E(G_3) \cup \cdots \cup E(G_{k-1})$ cover all edges in the induced subgraph $G'$ on $[2, n - 1]$, a contradiction to our induction hypothesis. Therefore, we must have $\{1, n\} \in E(G_1)$. If every edge in $E(G_1)$ meets $[1, d]$, then the $k - 2$ color classes $E(G_2) \cup \cdots \cup E(G_{k-1})$ cover the induced subgraph on $[d + 1, n]$, which is a contradiction by induction. It then must be that some edge in $E(G_1)$ avoids $[1, d]$, and the only possibility is $\{d + 1, n\} \in E(G_1)$, since no other edge avoids $[1, d]$ yet is close to $\{1, 2\}$ and $\{1, n\}$.

Suppose first that there is a cut $d + 1|d + 2$. Then every edge in $E(G_1)$ meets $[2, d] \cup \{n\}$, except $\{1, d + 1\}$ if it belongs to $E(G_1)$. Remove vertices $[2, d] \subseteq \{n\}$ and the edge $\{1, d + 1\}$, and replace them by a cut $1|d + 1$. The edges of this cut-graph on $n - d \geqq d(k + 2) + 3$ vertices have a coloring with just $k - 2$ colors induced by $G$, which contradicts our induction hypothesis.

Therefore there can be no cut between $d+1$ and $d+2$. They then form an edge in $G$ which is too far from $\{1, n\}$ to be color 1. Suppose, say, $\{d+1, d+2\} \in E(G_2)$. All edges meeting $[2, d]$ are color 1. So in order to be near $\{d+1, d+2\}$, every edge in $E(G_2)$ must meet the set $[d+1, p]$, where either $p = 2d+1$ or else $d+2 \leqq p \leqq 2d$ and $[1, p]$ is the section before a cut $p \mid p+1$.

Let $q = \min \{p, 2d\}$. Then every vertex in $[d+2, q]$ is distance at most $d-1$ from every edge in $E(G_2)$, so that $[d+2, q] \in H_2$. We may recolor every edge that meets $[d+2, q]$ by color 2. (The purpose of this recoloring is to get edges $\{1, j\}, j \in [d+2, q]$ out of $E(G_1)$.) Now the only edge in $E(G_1)$ that avoids $[2, d] \cup \{n\}$, if any do, is $\{1, d+1\}$. So as in the previous case, there is a $(k-2)$-coloring induced on the subgraph obtained by removing $[2, d] \cup \{n\}$ and inserting a cut $1 \mid d+1$. But there can be no such $(k-2)$-coloring of the subgraph, by the induction hypothesis, so we have a contradiction.

At this point we notice what the cut "bought" for us: an easy argument to reduce to the case that there is no cut near either end of $[1, n]$. We assume now that there are no cuts in $[1, d+1]$ or $[n-d, n]$. The edges $\{1, 2\}$ and $\{n-1, n\}$ belong to $E(G)$ and are different colors since $n \geqq d+3$. If $k = 2$, then there are at least two colors, and we are finished. Otherwise $k \geqq 3$, and suppose $\{1, 2\} \in E(G_1)$, $\{n-1, n\} \in E(G_2)$. As before, we find $[2, d] \in H_1$ and similarly $[n-d+1, n-1] \in H_2$. We may therefore recolor all edges meeting $[2, d]$ by color 1 and then all edges meeting $[n-d+1, n-1]$ by color 2. If $\{1, n\} \in E(G_3)$, say, then every edge in $E(G_3)$ meets 1 or $n$ so that by removing 1 and $n$ we get a $(k-2)$-coloring of a cut-graph on $n-2$ vertices, a contradiction to our induction hypothesis. Therefore, $\{1, n\}$ is color 1 or 2, say $\{1, n\} \in E(G_1)$. We now proceed exactly as before: We replace $[2, d] \cup \{n\}$ by a cut $1 \mid d+1$, eliminating all edges in $E(G_1)$. We carry out this replacement immediately if there is a cut between $d+1$ and $d+2$. Otherwise, if $\{d+1, d+2\} \in E(G)$, then it is a new color, say $\{d+1, d+2\} \in E(G_3)$; we recolor all edges meeting $[d+2, q]$ by color 3, with $q$ as above, and then replace $[2, d] \cup \{n\}$. We obtain a $(k-2)$-coloring of a cut-graph on at least $d(k-2)+3$ vertices, the same contradiction to our induction hypothesis as before.

In every case we have reached a contradiction, so at least $k$ colors are required to color $E(G)$, and the proposition follows. $\square$

This completes the proof of the lemma. $\square$

*Remarks.* A slightly stronger statement than the proposition can be proven by a similar argument. We can take away more edges every time there is a cut. Specifically, fix $d$ and a set $V = \{i_1, \cdots, i_a\} \subseteq [1, n]$. Then we may require that a *cut* $i_j \mid i_{j+1}$ omits not just one edge but omits every edge $\{i_{j+1-a}, i_{j+b}\}$ where $a \geqq 1$, $b \geqq 1$, $a+b \leqq d$, and no other cut separates $i_{j+1-a}$ and $i_{j+b}$. Return to the earlier example, $G = 1, 4 \mid 5, 6 \mid 8 \mid 9$ with $d = 3$. Then edges $\{1, 5\}$, $\{4, 6\}$, and $\{5, 8\}$ are also omitted besides $\{4, 5\}$, $\{6, 8\}$, and $\{8, 9\}$, as before. However, $\{6, 9\}$ is not omitted because two cuts intervene. The statement and proof of the proposition hold as above with this new definition of cut, even though the graphs have fewer edges in general.

It is also worth noting that the proof of the lemma is deceptively simple. Without introducing cut-graphs, there is no approach clearly available. Working on the ends at 1, 2 and $n-1$, $n$ also appears to be crucial to the argument.

For $d = 1$, the case of Kneser's graph of pairs, the proof of the lemma is rather easy. For the case originally proposed by Roberts, $d = 2$, the lemma states that $\chi(G(2k+1, 2)) \geqq k$. This graph $G(2k+1, 2)$ induces on the subset $\binom{\{1,3,5, \cdots, 2k+1\}}{2}$ of its vertex set a graph isomorphic to $G(k+1, 1)$. So we easily obtain

$\chi(G(2k+1,2)) \geqq \chi(G(k+1,1)) = k-1$. However, obtaining the desired lower bound of $k$ in this $d=2$ case seems to be essentially as difficult as the problem for general $d$.

## 4. Pair labellings with distance 2.

We now discuss the consequences of the findings above for the original problem of Roberts concerning pair labellings with distance 2 of graphs with given chromatic number. It follows from our main theorem that for graphs $G$ with $\chi(G) = k$ the pair labelling number, $r(G, 2)$, must be 2 if $k = 1$, and if $k \geqq 2$ it may assume any of the $k-1$ values in the range $[2k+1, 3k-1]$. The upper bound in this range, $3k-1$, is attained by the complete graph $K_k$. Of course any $k$-chromatic graph that contains $K_k$ also requires $3k-1$ labels. The lower bound in the range, $2k+1$, is attained by the "pair graph," $G(2k+1, 2)$, that was our main object of study.

As noted above, the 1-chromatic graphs $G$, which consist of isolated points, have $r(G, 2) = 2$, while the 2-chromatic graphs $G$, which are bipartite graphs with at least one edge, have $r(G, 2) = 5$. The 3-chromatic graphs have $r(G, 2) = 7$ or 8. It is curious that no graph has pair labelling number with value 3, 4, or 6.

We have seen that a graph $G$ has $r(G, 2) \leqq 7$ if and only if $G$ is homomorphic to the pair graph $G(7, 2)$ shown in Fig. 1. In the figure, $G(7, 2)$ is drawn to make evident its homomorphism to $C_5$. On the other hand, $C_5$ is immediately homomorphic to $G(7, 2)$ since it can be seen to be a subgraph. It follows that $r(G, 2) \leqq 7$ if and only if $G$ is homomorphic to $C_5$. Hence we have the following result.

PROPOSITION 4.1. *Suppose $G$ is a graph with $\chi(G) = 3$. Then $r(G, 2) = 7$ if and only if $G$ is homomorphic to $C_5$; else $r(G, 2) = 8$.*

Consider graphs $G$ with $\chi(G) = 3$. For instance, if $G$ is an odd cycle $C_{2k-1}$, $k \geqq 3$, then $r(G, 2) = 7$. If $G$ contains a triangle, then $r(G, 2) = 8$ since $G$ cannot be homomorphic to $C_5$. Next suppose the 3-chromatic graph $G$ is also triangle-free. It may still be that $G$ is not homomorphic to $C_5$, so that $r(G, 2) = 8$. In Fig. 2 we show an example of such a graph.

There is related work of interest on graph homomorphisms surveyed by Albertson [A]. Vesztergombi [V] characterized the 3-chromatic graphs that are homomorphic to $C_5$. A different but more powerful result was recently given by Gerards [G], who found an elegant obstruction characterization of nonbipartite graphs that are homomorphic to their shortest odd cycles. A related complexity result due to Maurer, Sudborough, and Welzl [MSW] is that the problem of determining whether a graph $G$ is homomorphic to $C_5$ is NP-complete in the size of $G$. Hence, determining whether $r(G, 2) \leqq 7$ is NP-complete. If $G$ is restricted to the class of 3-chromatic graphs, the complexity of this problem, i.e., the question of whether $r(G, 2)$ is 7 or 8, appears to be open. We suspect that this problem is NP-complete, so that there is no simple test to determine whether a 3-chromatic graph has $r(G, 2) = 7$.

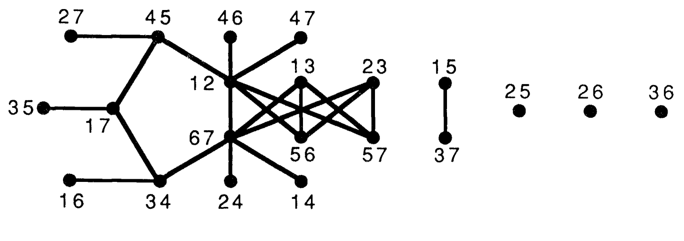


FIG. 1. *The graph $G(7, 2)$.*

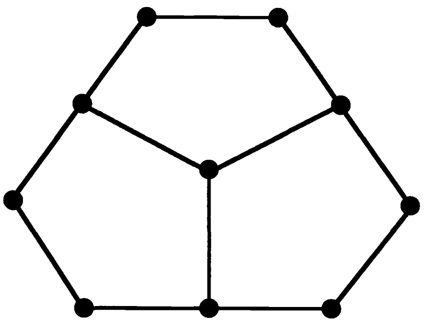


FIG. 2. *A triangle-free*, 3-*chromatic graph with* $r(G, 2) = 8$.

For general $k$, graphs $G$ with $\chi(G) = k$ and $r(G, 2) = j$, $2k + 1 \leqq j \leqq 3k - 1$, are, as in general, characterized as those $G$ with $\chi(G) = k$ that are homomorphic to $G(j, 2)$ but not to $G(j - 1, 2)$. For general $j$ this condition can be somewhat simplified since the graphs $G(j, 2)$ are homomorphic to some smaller graph $H_j$ as we saw above for $j = 7$. However, no nice description of suitable graphs $H_j$ for general $j$ is evident.

Another interesting family of graphs is the set of complements of odd cycles. For $k \geqq 3$, Lundgren [Lu] found that the complement of the $(2k - 1)$-cycle, $\bar{C}_{2k+1}$, has chromatic number $k$ and pair labelling number $3k - 2$, just one below the maximum value. Here is a labelling that achieves the minimum: Consecutive vertices receive pairs $\{1, 2\}$, $\{3, 4\}$, $\{4, 5\}$, $\{6, 7\}$, $\{7, 8\}$, $\cdots$, $\{3k - 3, 3k - 2\}$, $\{3k - 2, 1\}$. Indeed, we can show that this labelling is the *unique* labelling of $\bar{C}_{2k-1}$ on $[1, 3k - 1]$, up to isomorphism of the graph, but we omit the tedious details.

## 5. A Conjecture for *t*-labellings.

A weaker version of Theorem 2.1 holds for general $t$. It follows from Propositions 1.1 and 2.2 and from the discussion preceding Theorem 2.1.

THEOREM 5.1. *Let* $t, d, k \in \mathbb{Z}^+$. *Suppose the graph* $G$ *has* $\chi(G) = k$. *If* $k = 1$, *then* $r_t(G, d) = t$. *If* $k \geqq 2$, *then* $r_t(G, d) \in [d(k - 1) + 2t - 1, d(k - 1) + kt - k + 1]$. *The upper bound is attained by* $K_k$.

We have learned that Theorem 5.1 was independently discovered by Tesman [T] around the same time. For $t = 1$, the interval in the theorem consists of a single point, $d(k - 1) + 1$. For $t = 2$, Theorem 2.1 shows that all values in the interval are attained which is the main result of this paper. For $k = 2$, the interval again consists of a single point, $d + 2t - 1$. Therefore, it is reasonable to propose the following conjecture.

CONJECTURE 5.2. *All values in the interval in Theorem* 5.1 *are attained by suitable graphs* $G$.

If this conjecture holds, then the lower bound in Theorem 5.1 holds for some graph $G$ with $\chi(G) = k$. The same reasoning that we gave prior to the statement of Lemma 2.3 would then imply the following conjecture that generalizes Lemma 2.3.

CONJECTURE 5.3. *Let* $t, d, k \in \mathbb{Z}^+$. *Then* $\chi(G_t(d(k - 1) + 2t - 1, d)) \geqq k$.

Conjecture 5.3 appears to be slightly weaker than Conjecture 5.2 because the argument we gave to deduce Theorem 2.1 from Lemma 2.3 does not generalize for values of $t \geqq 3$. However, it may not be difficult to deduce Conjecture 5.2 from Conjecture 5.3.

For $d = 1$, Conjecture 5.3 is the famous theorem conjectured by Kneser [K] and first proved by Lovász [L]. The graphs $G_t(n, 1)$ are known as Kneser graphs and in other notation are denoted by $K(n, t)$ [FF] or by $KG_{t,n-2t}$ [S]. Bárány [B] gave a simpler proof. Schrijver [S] found a vertex-critical subgraph of the Kneser graph. Generalizations of Kneser's conjecture have been given recently in [FF] and [AFL]. All known proofs use topological methods relying on versions of the Borsuk–Ulam Theorem. Therefore, a purely graph-theoretic proof of our conjectures would be surprising. Perhaps there is a direct topological proof or one that follows from some of the work referenced above.

**Acknowledgments.** We are grateful to Fred Roberts and Midge Cozzens for calling this problem to our attention. We also thank the Institute for Mathematics and Its Applications at the University of Minnesota for its hospitality during our stay.

## REFERENCES

[A] M. O. Albertson, *Generalized Colorings*, *Discrete Algorithms and Complexity*, Academic Press, New York, 1987, pp. 35–49.

[AFL] N. Alon, P. Frankl, and L. Lovász, *The chromatic number of Kneser hypergraphs*, Trans. Amer. Math. Soc., 298 (1986), pp. 359–370.

[B] I. Bárány, *A short proof of Kneser's conjecture*, J. Combin. Theory Ser. A., 25 (1978), pp. 325–326.

[CR] M. B. Cozzens and F. S. Roberts, *T-colorings of graphs and the channel assignment problem*, Congr. Numer., 35 (1982), pp. 191–208.

[FF] P. Frankl and Z. Füredi, *Extremal problems concerning Kneser graphs*, J. Combin. Theory Ser. B, 40 (1986), pp. 270–284.

[G] A. M. H. Gerards, *Homomorphisms of graphs to odd cycles*, J. Graph Theory, 12 (1988), pp. 73–83.

[HN] P. Hell and J. Nesetril, *On the complexity of H-coloring*, preprint, (1986).

[K] M. Kneser, *Aufgabe* 300, Jber. Deutsch. Math. Verein, 58 (1955), p. 27.

[L] L. Lovász, *Kneser's conjecture, chromatic number, and homotopy*, J. Combin. Theory Ser. A., 25 (1978), pp. 319–324.

[Lu] J. R. Lundgren, IMA Workshop on Applications of Combinatorics and Graph Theory to the Biological and Social Sciences, Jan., 1988, personal communication.

[MSW] H. A. Maurer, I. H. Sudborough, and E. Welzl, *On the complexity of the general coloring problem*, Inform. and Control, 51 (1981), pp. 128–145.

[R1] F. S. Roberts, *T-colorings of graphs: recent results and open problems*, RUTCOR Research Report, Rutgers University, 1986.

[R2] ———, IMA Workshop on Applications of Combinatorics and Graph Theory to the Biological and Social Sciences, Jan., 1988, personal communication.

[R3] ———, *From garbage to rainbows: generalizations of graph colorings and their applications*, preprint, (1988).

[S] A. Schrijver, *Vertex-critical subgraphs of Kneser graphs*, Nieuw Archief Wisk. (3), 26 (1978), pp. 454–461.

[T] B. Tesman, Ph.D. thesis, Rutgers University, 1989, in preparation.

[V] K. Vesztergombi, *Some remarks on the chromatic number of the strong product of graphs*, Acta Cybernet., 4 (1978), pp. 207–212.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 500–507, November 1989

# CUTOFF POINT AND MONOTONICITY PROPERTIES FOR MULTINOMIAL GROUP TESTING*

F. K. HWANG† AND Y. C. YAO‡

**Abstract.** The classical binomial group testing problem studies plans to sort defectives from good items using a minimum number of group tests where a group test is a test applicable to any subset of items with a yes and no answer to the question of whether the subset contains any defectives. In a multinomial group testing problem, each item can be in one of $k$ ordered states and a group test on a subset always reveals the highest state of any item in the subset. Two properties known for binomial group testing are the cutoff point, which characterizes when individual testing is optimal, and the monotonicity, which states that the minimum expected number of tests increases in the probability of an item being defective. This paper studies the multinomial group testing problem and gives a new sufficient condition that guarantees individual testing is optimal. (A sufficient condition in Kumar [*SIAM J. Appl. Math.*, 19 (1970), pp. 340–350] is shown to be incorrect.) It is also shown that the monotonicity does not hold for the multinomial case.

**Key words.** group testing, cutoff points, monotonicity

**AMS(MOS) subject classifications.** 90B40, 90B25, 62L05, 60C05

## 1. Introduction.

Consider a set $N$ of $n$ independent items where each item has probability $q_j, j = 1, \cdots, k$, of being in state $j$ and $\sum_{j=1}^{k} q_j = 1$. The problem is to determine the state of each item through a sequence of tests. A test can be conducted on any subset $G$ of $N$ and the test outcome is $j$, written $f(G) = j$, if there exists an item in $G$ of state $j$, but no item of a higher state. This problem is known [2] as the multinomial group testing problem. The special case $k = 2$ is the well-known binomial group testing, or simply, group testing problem.

Let $q = \{q_1, \cdots, q_k\}$ and let $E(q, n, k)$ denote the expected number of tests required by an optimal testing plan for given $q$, $n$, and $k$. It is very difficult to determine $E(q, n, k)$ in general, even for the binomial case. Thus, it is helpful to learn as many properties of $E(q, n, k)$ as possible. Two such properties, cutoff point and monotonicity, have been well studied for the binomial case. The *cutoff point* is a characterization on $q$ such that $E(q, n, k) = n$, i.e., testing the items one at a time is optimal. *Monotonicity* refers to the property that it takes more tests to sort the items if more probability mass is distributed to higher states. More specifically, let $q$ and $p$ denote two probability distribution functions (pdfs) with no zero components. Then the conjecture states that $E(q, n, k) \leqq E(p, n, k)$ if $\sum_{i=m}^{k} q_i \leqq \sum_{i=m}^{k} p_i$ for each $m = 1, \cdots, k$.

For $k = 2$, Ungar [3] proved that $q_2 \geqq q_1^2$ is a necessary and sufficient condition for the cutoff point, while Yao and Hwang [4] recently proved the monotonicity property. Kumar attempted to extend Ungar's result to the trinomial case [1] and the general $k$ case [2]. However, Kumar's arguments contain some serious loopholes that cast doubts on his results. In this paper we give a new sufficient condition for individual testing to be optimal in multinomial group testing. We also show that the monotonicity property does not hold true for $k \geqq 3$.

## 2. Kumar's sufficient condition.

Following Ungar, we will represent an algorithm by a tree where each internal node is associated with a test, and we will consider only "reasonable" procedures in which no set $G$ will be tested if $G$ contains a subset $G' \subseteq G$ such that $f(G')$ is known before the test. Let $T$ be an algorithm in which there exists a

* Received by the editors April 11, 1988; accepted for publication (in revised form) March 31, 1989.
† AT&T Bell Laboratories, Murray Hill, New Jersey 07974.
‡ Colorado State University, Fort Collins, Colorado 80523.

test of size at least two. Let $B$ be a node associated with a test on the set $G$ of size $g \geqq 2$, but all tests following $B$ are of size one. (Such a $B$ must exist, and the associated test is not a final test because we assume $T$ is reasonable.) Let $T'$ be the algorithm obtained from $T$ by substituting $G - a$ for $G$ at $B$ where $a$ is some item in $G$. We then test $a$ and follow the branch of $T$ associated with the output $f(G)$, which we can figure out from $f(G - a)$ and $f(a)$, except the test on $a$ is skipped if $f(G - a) = k$. $T'$ will skip any test of $T$ if its outcome can be deduced from $f(G - a)$ and $f(a)$.

Kumar [2] purported to show that for $k \geqq 3$ and $n \geqq 2$ if

$$\sum_{j=k-1}^{k} q_j \sum_{i=1}^{j} q_i + \sum_{j=2}^{k-2} q_j(q_1 + q_j) \geqq q_1^2,$$

then $T'$ is at least as good as $T$; hence, individual testing is optimal. However, this conclusion is invalidated by the following loopholes contained in his proof.

(i) He uses the unconditional probability $q_j$ for an item that may have already been included in some previous tests.

(ii) For $|G| \geqq 3$ he requires $a$ not to be the last item of $G$ to be tested under $T$ along either the branch $f(G) = k - 1$ or the branch $f(G) = k$. But the last item on either branch may still depend on the outcomes of future tests. Hence, there may not exist an item $a$ satisfying the requirement.

(iii) He ignores the possibility that an item $b \in G$ can skip an individual test if there exists a previous test containing $b$ and $b$ is the unique item of the highest state in that test.

There may be an easy correction for (i), but not for (ii) and (iii). In fact, we will give an example to show that Kumar's condition does not guarantee $T'$ to be at least as good as $T$.

*Example.* Let $N = \{A, B, C\}$ and $k \geqq 3$. Let $T$ and $T'$ be as shown in Fig. 1 where $T_i$'s are subplans of $T$. In Fig. 1, each node is labeled by the set of items to be tested and each outlink is labeled by one of the $k$ states $1, 2, \cdots, k$, indicating the outcome of the test.

Note that $T'$ differs from $T$ only when $f(ABC) = f(C) = 2$. The comparison between $T$ and $T'$ reduces to the case $k = n = 2$ where items $A$ and $B$ are each in state 2 with probability $q_2/(q_1 + q_2)$. The well-known result of Ungar implies that $T'$ is at least as good as $T$ if and only if $q_2/(q_1 + q_2) \geqq \{q_1/(q_1 + q_2)\}^2$, or equivalently, $q_2(q_1 + q_2) \geqq q_1^2$. But clearly this condition is not implied by Kumar's condition as stated earlier.

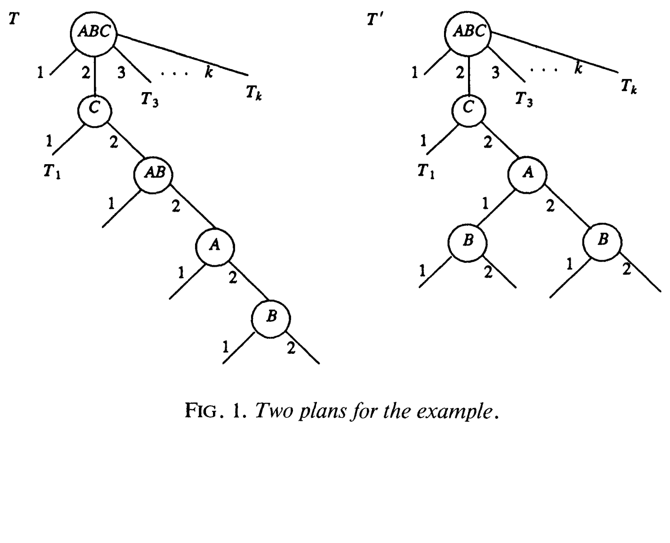


FIG. 1. *Two plans for the example.*

**3. Sufficient conditions.** In what follows, we assume that $q_i > 0$ for $i = 1, \cdots, k$. (If some $q_i$ equals zero, then the problem reduces to a smaller $k$ case.) Let $E(n, q_1, \cdots, q_k)$ denote the minimum expected number of tests required to classify $n$ items. Define

$$Q_k = \{(q_1, \cdots, q_k): q_i > 0, i = 1, \cdots, k; q_1 + \cdots + q_k = 1;$$
$$E(n, q_1, \cdots, q_k) = n \text{ for all } n\}.$$

In other words, $Q_k$ consists of those probability distributions under which the individual testing algorithm is optimal for every $n$.

THEOREM 1. *$(q_1, \cdots, q_k) \in Q_k$ if $q_2(q_1 + q_2) \geqq q_1^2$.*

*Proof.* Suppose $T$ is a reasonable algorithm which includes a test of size $\geqq 2$. Let $G$ be a test of size $m \geqq 2$ associated with a node $B$ in $T$ such that every test below $B$ is of size 1. We will construct a new algorithm $T'$ so that the expected number of tests under $T'$ is less than that under $T$ if $q_2(q_1 + q_2) > q_1^2$. (By continuing the type of transformation used to produce $T'$ until all tests are of size 1, individual testing must be optimal if $q_2(q_1 + q_2) > q_1^2$. So, $E(n, q_1, \cdots, q_k) = n$ if $q_2(q_1 + q_2) > q_1^2$. Since $E(n, q_1, \cdots, q_k)$ is continuous in the $q$'s, we have $E(n, q_1, \cdots, q_k) = n$ if $q_2(q_1 + q_2) \geqq q_1^2$. This shows that $(q_1, \cdots, q_k) \in Q_k$ if $q_2(q_1 + q_2) \geqq q_1^2$.) For every $w \in G$, let $F(w) = k$ if $w$ is not included in any test above $B$; and $F(w) = j$ if $w$ is included in a test (above $B$) with outcome $= j$ and included in no test with outcome $< j$. (Note that $F$ is known before $G$ is tested.) So, $f(w) \leqq F(w)$. Let $k_1 = \max\{F(w): w \in G\}$ and choose $a \in G$ so that $F(a) = k_1$. Let $k_2 = \max\{F(w): w \in G - a\}$. Clearly, $2 \leqq k_2 \leqq k_1$ since $T$ is reasonable. Now, consider $T$ and $T'$ as shown in Fig. 2. Here $T_i'$ is the same as $T_i$ except that tests with known outcome are skipped. Note that, under the new plan, if $f(a) \geqq k_2$, then $f(G) = f(a)$ since $k_2 = \max\{F(w): w \in G - a\}$. This is why $G - a$ is not tested when $f(a) \geqq k_2$.

Let $s$ be a sample which reaches $B$. Let $O(s)$ and $N(s)$ denote the number of tests required to classify $s$ under the old and new plans, respectively. Clearly, $N(s) \leqq O(s) + 1$. Thus, $N(s) \leqq O(s)$ if one of the following conditions is satisfied:

(i) $f(a) \geqq k_2$,

(ii) $f(a) < k_2$, $f(G - a) > 1$, and $f(a) \leqq f(G - a)$. (Note that under the old plan, under condition (ii), $a$ will have to be tested individually since, up to and including $G$, every test which contains $a$ must contain another item whose state is $\geqq \max(f(a), 2)$.)

(iii) $f(G - a) = 1 < f(a) < k_2$.

Therefore, the set of $s$ satisfying $N(s) = O(s) + 1$ is the union of $S_1$ and $S_2$ where

$$S_1 = \{s: s \text{ reaches } B \text{ and } f(G) = 1\}$$

$$S_2 = \{s: s \text{ reaches } B, 1 < f(G - a) < f(a) < k_2 \text{ and } N(s) = O(s) + 1\}.$$

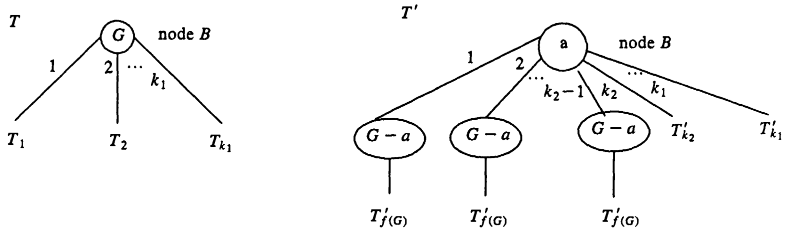


FIG. 2. *Two plans for Theorem* 1.

We will show that for each sample in $S_1$ or $S_2$, there is another sample for which the new plan saves at least one test over the old plan. Calculation of the relative probabilities of these samples will then indicate that the new plan is better than the old one.

We claim that $S_1$ is nonempty. To see this, note that the function $F$ is defined for $w \in G$, and it can be defined for $w \notin G$ in a similar way. Consider the sample $s$ satisfying $f(w) = F(w)$ for $w \notin G$ and $f(w) = 1$ for $w \in G$. Since $T$ is reasonable, it is not difficult to show that $s \in S_1$. The set $S_2$ is possibly empty. We assume $S_2$ is nonempty. (Otherwise, the following discussion on $S_2$ is unnecessary.) Fix $s \in S_2$. Let

$$G'(s) = \{w \in G - a : F(w) \geqq f(a)\},$$

which is nonempty because $s \in S_2$ and $\max\{F(w) : w \in G - a\} = k_2 > f(a)$. Clearly, under the old plan, every item $w$ in $G'(s)$ must be tested individually since up to and including $G$, every test that contains $w$ must contain another item whose state is $> f(w)$. Let $b = b(s)$ be the last item in $G'(s)$ to be tested individually. (After $b$ is tested, $a$ will be classified since, at this point, it is known that the value of $f(G - a)$ must be less than $f(G)$). Let $s_a^b$ be the same as $s$ except that the states of $a$ and $b$ are interchanged. Note that $\Pr(s_a^b) = \Pr(s)$. We introduce the notation $d(a|s)$ to denote the state of $a$ for sample $s$ in order to avoid confusion. Clearly, under the old plan, $s_a^b$ will reach the test of $b$ since $s_a^b$ and $s$ have the same outcomes for all previous tests. For $s_a^b$, both $a$ and $b$ have to be tested individually under the old plan. But under the new plan, for the sample $s_a^b$, right before $b$ is tested, the value of $f(G - a|s_a^b)$ is known and equals $f(a|s)$, the value of $f(w|s_a^b)$ is known and less than $f(a|s)$ for $w \in G'(s) - b$, and the value of $f(z|s_a^b)$ is possibly unknown but we know it must be $\leqq F(z) < f(a|s)$ for $z \in G - a - G'(s)$. Therefore, the value of $f(b|s_a^b)$ is known to be $f(a|s)$, and the test of $b$ is skipped under the new plan. So, for every $s \in S_2$, we have $O(s_a^b) \geqq N(s_a^b) + 1$.

Next, fix $s \in S_1$. We consider three separate cases.

*Case* (i). $m = 2$. Let $x$ denote the item other than $a$ in $G$. Let $s_{ax}$ be a sample which is the same as $s$ except that $f(a|s_{ax}) = f(x|s_{ax}) = 2 \leqq k_2$. For $s_{ax}$, every test $H$ before $G$ which contains $a$ or $x$ must contain another item with state $\geqq 2$ because, otherwise, $f(H|s) = 1$, which is a contradiction with the assumption of $T$ being reasonable. So, for $s_{ax}$, under the old plan, both $a$ and $x$ have to be tested individually. That is, $O(s_{ax}) = N(s_{ax}) + 1$. Now, for $s_{ax}$, let $y$ be that one of $a$ and $x$ which is tested first when the old plan is applied to $s_{ax}$. Let $s_y$ be the same as $s$ except that $f(y|s_y) = 2$. Clearly, $O(s_y) = N(s_y) + 1$. Therefore, for the case $m = 2$, the expected number of tests saved by the new plan is at least

$$\sum_{s \in S_1} \{\Pr(s_{ax}) + \Pr(s_y) - \Pr(s)\} + \sum_{s \in S_2} \{\Pr(s_a^b) - \Pr(s)\}$$

$$= \sum_{s \in S_1} \Pr(s) \left\{\frac{q_2^2}{q_1^2} + \frac{q_2}{q_1} - 1\right\} > 0 \quad \text{if } q_2(q_1 + q_2) > q_1^2.$$

It should be remarked that every sample in the first sum satisfies $f(G) \leqq 2$ and every sample in the second sum satisfies $f(G) \geqq 3$. Therefore, no sample is counted twice in the above expression.

*Case* (ii). $m \geqq 3$ and $k_2 \geqq 3$. Again fix $s \in S_1$. Let $s_a$ be the same as $s$ except that $f(a|s_a) = 2$. There is a net saving of at least $m - 2$ when we test $s_a$ under the new plan, since every item in $G - a$ has to be tested individually under the old plan.

Next, let $z$ be the last item of $G - a$ to be tested when we apply the old plan to $s_a$. Let $s_{az}$ be the same as $s$ except that $f(a|s_{az}) = f(z|s_{az}) = 2$. Then $z$ will also be the last element of $G - a$ to be tested when we test $s_{az}$ under the old plan, since all previous test outcomes are the same for $s_a$ and $s_{az}$. When we test $s_{az}$ under the new plan, we can obviously skip the test of $z$. Also, $a$ has to be tested when the old plan is applied to $s_{az}$. So, $O(s_{az}) \geqq N(s_{az}) + 1$.

Therefore, for the case $m \geqq 3$ and $k_2 \geqq 3$, the expected number of tests saved by the new plan is at least

$$\sum_{s \in S_1} \{\Pr(s_{az}) + (m-2)\Pr(s_a) - \Pr(s)\} + \sum_{s \in S_2} \{\Pr(s_a^b) - \Pr(s)\}$$

$$\geqq \sum_{s \in S_1} \Pr(s)\left\{\frac{q_2^2}{q_1^2} + \frac{q_2}{q_1} - 1\right\} > 0, \quad \text{if } q_2(q_1+q_2) > q_1^2.$$

*Case* (iii). $m \geqq 3$ and $k_2 = 2$. Fix $s \in S_1$ and let $s_a$ and $s_{az}$ be defined as in Case (ii). Note that for $s_a$, $G - a$ is not tested under the new plan since $f(a \mid s_a) = 2 = k_2$. We have $O(s_{az}) \geqq N(s_{az}) + 1$. Consider $S_3 = \{s \in S_1 :$ item $a$ of sample $s_a$ is never tested individually under the old plan$\}$. So, for $s \in S_1 - S_3$, we have $O(s_a) \geqq N(s_a) + 1$. Recall that $z$ is the last item in $G - a$ to be tested when the old plan is applied to $s_a$. Let $s_z$ be the sample satisfying $f(z \mid s_z) = 2$ and $f(w \mid s_z) = f(w \mid s)$, $w \neq z$. Under the old plan, for $s \in S_3$, $s_z$ will reach the test of $z$ since all previous test outcomes are identical for $s_a$ and $s_z$. But for $s_z$, the test of $z$ can be skipped under the new plan since it is known that $f(G - a - z \mid s_z) = 1$ and $f(G - a \mid s_z) = 2$. In addition, under the old plan, for $s_z$, $a$ still has to be tested individually. So, for $s \in S_3$, we have $O(s_z) \geqq N(s_z) + 1$.

Therefore, for the case $m \geqq 3$ and $k_2 = 2$, the expected number of tests saved by the new plan is at least

$$\sum_{s \in S_3} \{\Pr(s_{az}) + \Pr(s_z) - \Pr(s)\}$$

$$+ \sum_{s \in S_1 - S_3} \{\Pr(s_{az}) + \Pr(s_a) - \Pr(s)\} + \sum_{s \in S_2} \{\Pr(s_a^b) - \Pr(s)\}$$

$$= \sum_{s \in S_1} \Pr(s)\left\{\frac{q_2^2}{q_1^2} + \frac{q_2}{q_1} - 1\right\} > 0 \quad \text{if } q_2(q_1+q_2) > q_1^2,$$

completing the proof. □

We will extend the sufficient condition in Theorem 1 to a more general form. The vehicle of the extension is the following theorem.

THEOREM 2. *For* $2 \leqq k < l$, *if* $(q_1, \cdots, q_l)$ *is such that*

$$q_i > 0, \quad i = 1, \cdots, l, \quad q_1 + \cdots + q_l = 1$$

*and*

$$\left(\sum_{i=1}^{l-k+1} q_i, q_{l-k+2}, \cdots, q_{l-1}, q_l\right) \in Q_k,$$

*then*

$$(q_1, \cdots, q_l) \in Q_l.$$

*Proof.* We have to show $E(n, q_1, \cdots, q_l) = n$. Consider a new problem in which we want to classify each item into "$k$" categories: an item is in category $l - k + 1$ if its state is $\leqq l - k + 1$; an item is in category $j$ $(j = l - k + 2, \cdots, l)$ if its state is $j$. Consider three different machines:

Machine $M_1$: the outcome of a test is $l-k+1$ if the state of every item in the test is $\leqq l-k+1$; is $j$ ($j=l-k+2,\cdots,l$) if some item is in state $j$ and no item is in state $>j$.

Machine $M_2$: the outcome of a test is $j$ ($j=1,\cdots,l$) if some item is in state $j$ and no item is in state $>j$.

Machine $M_3$: the same as $M_2$ except that if the outcome of a test is $\leqq l-k+1$, then the information about the state of every item in the test is also provided.

Let $\tilde{E}_r(n,q_1,\cdots,q_l)$ denote the minimum expected number of tests for the new problem using machine $M_r$, $r=1,2,3$. Clearly, $\tilde{E}_3 \leqq \tilde{E}_2 \leqq \tilde{E}_1$. Also, $\tilde{E}_2(n,q_1,\cdots,q_l) \leqq E(n,q_1,\cdots,q_l)$ and

$$\tilde{E}_1(n,q_1,\cdots,q_l)=E(n,\textstyle\sum_{i=1}^{l-k+1}q_i,q_{l-k+2},\cdots,q_l)=n,$$

since the new problem with machine $M_1$ is equivalent to the original problem with $k$ states and probability distribution $(\sum_{i=1}^{l-k+1}q_i,q_{l-k+2},\cdots,q_l)$. For machine $M_3$, there must exist an optimal algorithm in which no test includes an item whose state is known. So, under this algorithm, when a test has outcome $\leqq l-k+1$, the state of each item in this test is known so that these items will not be included in any future test. In addition, the information about the states of these items should not affect the choice of the future tests since the conditional distribution of the states of the unclassified items does not depend on this information. Therefore, this algorithm can also be applied to machine $M_1$. This shows $\tilde{E}_3=\tilde{E}_1$. So,

$$n=\tilde{E}_1=\tilde{E}_3\leqq\tilde{E}_2\leqq E(n,q_1,\cdots,q_l).$$

So, $E(n,q_1,\cdots,q_l)=n$, completing the proof. □

We now give an improved sufficient condition.

THEOREM 3. *For $k\geqq 2$, and $q_i>0$, $i=1,\cdots,k$, if there exists $2\leqq i\leqq k$ such that*

$$\frac{\sqrt{5}-1}{2}\leqq\frac{q_i}{\sum_{j=1}^{i-1}q_j},$$

*then $(q_1,\cdots,q_k)\in Q_k$.*

*Proof.* We proceed by induction on $k$. The theorem holds for $k=2$ as a consequence of Ungar [3]. Suppose the theorem holds for $2,\cdots,l-1$ ($l\geqq 3$). Let $i$ be such that $2\leqq i\leqq l$ and

$$\frac{\sqrt{5}-1}{2}\leqq\frac{q_i}{\sum_{j=1}^{i-1}q_j}<\infty.$$

If $i=2$, it follows from Theorem 1 that $(q_1,\cdots,q_l)\in Q_l$. If $i>2$, consider $q'_1=q_1+q_2$, $q'_2=q_3,\cdots,q'_{l-1}=q_l$. Since

$$\frac{\sqrt{5}-1}{2}\leqq\frac{q'_{i-1}}{\sum_{j=1}^{i-2}q'_j}<\infty,$$

$(q'_1,\cdots,q'_{l-1})\in Q_{l-1}$ by the induction hypothesis. By Theorem 2,

$$(q_1,\cdots,q_l)\in Q_l. \qquad \square$$

The condition "$q_2(q_1+q_2)\geqq q_1^2$" of Theorem 1 implies that $(\sqrt{5}-1)/2\leqq q_2/q_1$, which implies the condition of Theorem 3. But the condition of Theorem 3 does not imply that of Theorem 1. For example, for $k = 3$, $(q_1, q_2, q_3) = (0.4, 0.1, 0.5)$ satisfies the condition of Theorem 3 but does not satisfy the condition of Theorem 1.

**4. Monotonicity.** We first give a counterexample to the monotonicity property for $k = 3$. Let $N = \{A, B\}$. Let $q$ and $p$ be two distributions such that $q_1 = 1 - \varepsilon$, $q_2 = \varepsilon - \varepsilon^3$, $q_3 = \varepsilon^3$, and $p_1 = 1 - \varepsilon$, $p_2 = p_3 = \varepsilon/2$ where $\varepsilon$ is a sufficiently small positive number. Clearly $\sum_{i=m}^3 q_i \leqq \sum_{i=m}^3 p_i$ for $m = 1, 2, 3$. For sufficiently small $\varepsilon$, an optimal plan under either $p$ or $q$ must start with a test on $N$, since the minimum expected number of tests required must be $1 + O(\varepsilon)$ and a plan starting with a test on a proper subset of $N$ needs two tests. Let $E_p$ and $E_q$ denote the expected number of tests under $p$ and $q$, respectively. Then

$$E_q = 1 + 3\varepsilon - \varepsilon^2 + O(\varepsilon^3)$$

and

$$E_p = 1 + 3\varepsilon - 1.25\varepsilon^2 < E_q.$$

An optimal plan for $N = \{A, B\}$ for both distributions $q$ and $p$ (for sufficiently small $\varepsilon$) is given in Fig. 3.

This counterexample can be extended to the case $k > 3$ as follows. Again, let $N = \{A, B\}$. For a sufficiently small positive $\varepsilon$, let $q$ and $p$ be two distributions such that

$$q_1 = (1-\varepsilon^3)(1-\varepsilon), \quad q_2 = (1-\varepsilon^3)(\varepsilon - \varepsilon^3), \quad q_3 = (1-\varepsilon^3)\varepsilon^3$$

$$q_i = \varepsilon^3/(k-3), \qquad i = 4, \cdots, k;$$

$$p_1 = (1-\varepsilon^3)(1-\varepsilon), \qquad p_2 = p_3 = (1-\varepsilon^3)\varepsilon/2,$$

$$p_i = \varepsilon^3/(k-3), \qquad i = 4, \cdots, k.$$

An optimal plan for either $p$ or $q$ must start with a test on $N$. The minimum expected number of tests under $q$ and $p$ are, respectively,

$$E_q = 1 + 3\varepsilon - \varepsilon^2 + O(\varepsilon^2)$$

$$E_p = 1 + 3\varepsilon - 1.25\varepsilon^2 + O(\varepsilon^3) < E_q.$$

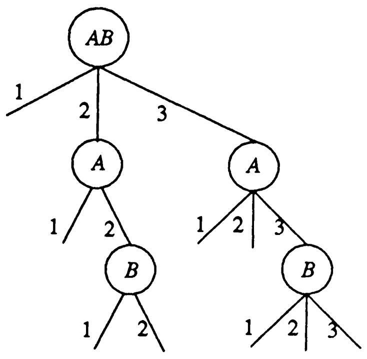


FIG. 3. *An optimal plan for $N = \{A, B\}$.*

Also note that

$$\sum_{i=m}^{k} q_i \leqq \sum_{i=m}^{k} p_i, \qquad m=1,\cdots,k.$$

Therefore, the monotonicity property does not hold for $k \geqq 3$.

**5. Conclusions.** Although binomial group testing ($k = 2$) has given rise to a large body of literature over the years, not much is known about the optimal plans and the minimum expected numbers of tests. The only two known properties are the cutoff point and monotonicity. In this paper we show that there is no straightforward extension of these two properties to the multinomial case. In fact monotonicity simply does not hold for $k \geqq 3$. Furthermore, we show that a previous attempt by Kumar of extending Ungar's arguments on cutoff points to the multinomial case contains some serious loopholes. We offer a different argument which leads to a new sufficient condition. Though this condition is not a necessary one, the example (for any $k \geqq 3$) given in § 2 showed that this is the best condition we can get by adopting Ungar's approach which concerns only a final group of size greater than one. It seems that, to completely characterize the condition for the cutoff point, we must find a way to analyze groups, other than the last one, of sizes greater than one.

## REFERENCES

[1] S. KUMAR, *Group-testing to classify all units in a trinomial sample*, Studia Sci. Math. Hungar., 5 (1970), pp. 229–247.

[2] ———, *Multinomial group-testing*, SIAM J. Appl. Math., 19 (1970), pp. 340–350.

[3] P. UNGAR, *The cutoff point for group testing*, Comm. Pure Appl. Math., 13 (1960), pp. 49–54.

[4] Y. C. YAO AND F. K. HWANG, *A fundamental monotonicity in group testing*, SIAM J. Disc. Math. 1, (1988), pp. 256–259.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 508-523, November 1989

# AVERAGE PERFORMANCE OF HEURISTICS FOR SATISFIABILITY*

RAJEEV KOHLI† AND RAMESH KRISHNAMURTI‡

**Abstract.** Distribution-free tight lower bounds on the average performance ratio for random search, for a greedy heuristic and for a probabilistic greedy heuristic are derived for an optimization version of satisfiability. On average, the random solution is never worse than $\frac{1}{2}$ of the optimal, regardless of the data-generating distribution. The lower bound on the average greedy solution is at least $\frac{1}{2}$ of the optimal, and this bound increases with the probability of the greedy heuristic selecting the optimal at each step. In the probabilistic greedy heuristic, probabilities are introduced into the search strategy so that a decrease in the probability of finding the optimal solution occurs only if the nonoptimal solution becomes closer to the optimal. Across problem instances, and regardless of the distribution giving rise to data, the minimum average value of the solutions identified by the probabilistic greedy heuristic is no less than $\frac{2}{3}$ of the optimal.

**Key words.** satisfiability, greedy heuristics, probabilistic heuristics, average performance

**AMS(MOS) subject classification.** 68Q25

## 1. Introduction.

This paper examines the average performance of random search, of a greedy heuristic and of a probabilistic version of a greedy heuristic for an optimization version of satisfiability. We derive tight lower bounds on the average performance of each heuristic. The analysis assumes no specific data-generating distributions and therefore is valid for all distributions.

A variety of analytic approaches have recently been pursued to analyze the average-case performance of heuristics. These include representing the execution of algorithms by Markov chains (Coffman, Leuker, and Rinnooy Kan [5]), obtaining the performance bound for a more tractable function that dominates the performance of the heuristic for each problem instance (Bruno and Downey [3], Boxma [2]), and obtaining the performance bound for a simpler, more easily analyzed heuristic which dominates the heuristic of interest for each problem instance (Csirik et al. [6]). Bounds that hold for most problem instances have also been employed to obtain asymptotic bounds for the average-case performance of various heuristics (Bentley et al. [1] and Coffman and Leighton [4]). A number of results from applied probability theory have been used for average-case analyses by Frenk and Rinnooy Kan [10], Karp, Luby, and Marchetti-Spaccamela [14], Shor [17], and Leighton and Shor [15]. The vast majority of these approaches begins by assuming independent, identically distributed data from a given density function. The subsequent analyses are often difficult, and one rarely finds an explicit formula for the quantity of interest. One reason for this is that conditional probabilities arise in the analyses, and after a sufficient number of steps, the conditioning can make the analyses formidable. Appropriate choice of distributional assumptions also is difficult, as are inferences regarding the robustness of results for a given distribution to other distributions.

A well-known algorithm for solving satisfiability is the Davis–Putnam Procedure (Davis, Logemann, and Loveland [7]). Goldberg, Purdom, and Brown [11], and Franco and Paull [9] have analyzed the average-case *complexity* of variants of this procedure for solving satisfiability. Johnson [13] considers an optimization version of satisfiability, called maximum satisfiability, proposes two heuristics for solving the maximum satisfiability problem, and proves tight worst-case bounds on the performances of these heu-

* Received by the editors December 15, 1988; accepted for publication February 28, 1989. This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant OGP0036809.

† Graduate School of Business, University of Pittsburgh, Pittsburgh, Pennsylvania 15260.
‡ School of Computing Science, Simon Fraser University, Burnaby, British Columbia, V5A 1S6, Canada.

ristics. One of these heuristics is the greedy heuristic that we use in this paper. If each clause contains at least $l$ variables, Johnson [13] shows a tight worst-case bound of $l/(l+1)$ for the greedy heuristic. Since we consider the most general optimization version of satisfiability, where unary clauses (clauses with just one variable) are allowed, this bound reduces to $\frac{1}{2}$. As one of our results, we derive this bound using a different approach. Lieberherr and Specker [16] provide the best possible polynomial-time algorithm for the maximum satisfiability problem where unary clauses are allowed, but the set of clauses must be 2-satisfiable, i.e., any two of the clauses are simultaneously satisfiable. The lower bound obtained for their algorithm is 0.618.

In the present analyses, we consider the lower bound of the average performance making no assumption regarding the data-generating distribution. For two of the three procedures (random search and the probabilistic greedy heuristic), we also make no assumption regarding the independence of data. For the third (the greedy heuristic), we assume independence, but only in a certain "aggregate" sense, which we discuss later. Each of the bounds we obtain is tight. Our central results are as follows. Random search, which has an arbitrarily bad performance in the worst case, provides solutions that, on average, are never worse than $\frac{1}{2}$ of the optimal. The greedy heuristic can potentially improve on this performance. Although the lower bound on its average performance ratio can be $\frac{1}{2}$ of the optimal, this lower bound increases with the probability of the heuristic selecting the optimal at each step. A probabilistic algorithm related to the greedy heuristic is then described. The probabilities are introduced into the search strategy so that a decrease in the probability of finding the optimal solution occurs only if the non-optimal solution becomes closer to the optimal. The search probabilities are not fixed a priori but exploit the structure of the data to force a trade-off for every problem instance. Across problem instances, and regardless of the distribution giving rise to the data, the average performance of the algorithm is never less than $\frac{2}{3}$ of the optimal.

Section 2 describes the maximum satisfiability problem, the random search procedure, and obtains a tight lower bound on its average performance. Section 3 introduces the greedy heuristic, derives its worst-case bound, and a tight lower bound on its average performance. Section 4 describes the probabilistic greedy heuristic and derives a tight lower bound on its average performance.

## 2. The Msat problem.

Consider the following optimization version of satisfiability: given $n$ clauses, each described by a disjunction of a subset of $k$ variables or their negations, find a truth assignment for the variables that maximizes the number of clauses satisfied. The above problem, which is the most general version of maximum satisfiability, is NP-complete (Johnson [13]). We call this Msat.

We use the following tabular representation of Msat. For a problem involving $n$ clauses and $k$ variables, construct a table $T_k$ with $n$ rows and $2k$ columns. The $i$th row is associated with clause $i$, $i = 1, \cdots, n$. A pair of columns, $u_j$, $\bar{u}_j$, is associated with the $j$th variable, $j = 1, \cdots, k$. Let $t_{ij}$ denote the entry in the cell identified by row $i$ and column $u_j$, and let $\bar{t}_{ij}$ denote the entry in the cell identified by row $i$ and column $\bar{u}_j$. For $i = 1, \cdots, n, j = 1, \cdots, k$, define

$$\begin{cases} t_{ij}=1, \bar{t}_{ij}=0, & \text{if clause } i \text{ contains variable } j, \\ t_{ij}=0, \bar{t}_{ij}=1, & \text{if clause } i \text{ contains the negation of variable } j, \\ t_{ij}=0, \bar{t}_{ij}=0, & \text{if clause } i \text{ contains neither variable } j \text{ nor its negation.} \end{cases}$$

A truth assignment for satisfiability results in the $j$th variable being assigned a $T$ (*True*) or an $F$ (*False*), $j = 1, \cdots, k$. This corresponds to selecting either column $u_j$

(if the $j$th variable is assigned a $T$) or $\bar{u}_j$ (if the $j$th variable is assigned an $F$), $j = 1, \cdots, k$, for Msat. Consequently, selecting $u_j$ or $\bar{u}_j$ for each $j$, $j = 1, \cdots, k$, such that the maximum number of rows in these columns have at least one 1, corresponds to solving Msat for $T_k$; i.e., finding a truth assignment that maximizes the number of clauses satisfied.

Let $T(u_k)(T(\bar{u}_k))$ denote the table obtained by deleting from $T_k$ all rows with a 1 in column $u_k(\bar{u}_k)$, and deleting *both* $u_k$ and $\bar{u}_k$. Let the resulting table be denoted $T_{k-1}$. That is,

$$T_{k-1} = \begin{cases} T(u_k), & \text{if column } u_k \text{ is chosen from } T_k, \\ T(\bar{u}_k), & \text{if column } \bar{u}_k \text{ is chosen from } T_k. \end{cases}$$

In general, let $T(u_j)(T(\bar{u}_j))$ denote the table obtained by deleting from $T_j$ all rows with a 1 in column $u_j(\bar{u}_j)$, and deleting both $u_j$ and $\bar{u}_j$, $j = 1, \cdots, k$. Let the resulting table be denoted $T_{j-1}$. That is,

$$T_{j-1} = \begin{cases} T(u_j), & \text{if column } u_j \text{ is chosen from } T_j, \\ T(\bar{u}_j), & \text{if column } \bar{u}_j \text{ is chosen from } T_j. \end{cases}$$

Let $x_j$ denote the number of 1's in $u_j$ and let $n_j$ denote the total number of 1's across columns $u_j$ and $\bar{u}_j$ in table $T_j$, $j = 1, \cdots, k$. Without loss of generality, assume that the columns $u_j$, $j = 1, \cdots, k$, comprise the optimal solution for Msat described by $T_k$. Let $m_k$ denote the optimal solution to Msat described by $T_k$. In general, let $m_j$ denote the value of the optimal solution to Msat described by $T_j$, $j = 1, \cdots, k$. Also, let $a_j(\bar{a}_j)$ denote the value of the optimal solution to Msat described by $T(u_j)(T(\bar{u}_j))$, $j = 1, \cdots, k$. That is,

$$m_{j-1} = \begin{cases} a_j, & \text{if column } u_j \text{ is chosen from } T_j, \\ \bar{a}_j, & \text{if column } \bar{u}_j \text{ is chosen from } T_j. \end{cases}$$

*Example* 1. Consider a problem consisting of three variables $x_1$, $x_2$, and $x_3$ and seven clauses given by $x_1 + x_2 + x_3$, $\bar{x}_1 + x_2 + \bar{x}_3$, $x_1 + \bar{x}_2 + \bar{x}_3$, $\bar{x}_1 + x_3$, $x_2 + \bar{x}_3$, $x_1 + x_2$, and $\bar{x}_1$. Table $T_3$ for this problem is given in Fig. 1.

For the table $T_3$ above, table $T(u_3)$ is obtained by deleting rows 1 and 4 and the columns $u_3$ and $\bar{u}_3$. Table $T_2 = T(u_3)$ is given in Fig. 2. Similarly, we can obtain table $T(\bar{u}_3)$ by deleting rows 2, 3, and 5 and the columns $u_3$ and $\bar{u}_3$. Table $T_2 = T(\bar{u}_3)$ is given in Fig. 3.

Consider a random procedure that selects column $u_j$ or $\bar{u}_j$ with probability $\frac{1}{2}$, $j = 1, \cdots, k$. The procedure can easily be seen to select an arbitrarily bad solution in the

| Row | $u_3$ | $\bar{u}_3$ | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 0 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 |

FIG. 1. *Table $T_3$ for Example* 1.

| Row | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |

FIG. 2. *Table* $T_2 = T(u_3)$ *for Example* 1.

worst case. But how poorly does it do on the average? That is, if *any* data-generating mechanism is used to construct instances of Msat, and if for each problem instance a random solution is selected, what is the average value of the ratio of the random solution value to the optimal solution value? Theorem 1 shows that it is never less than $\frac{1}{2}$. That is, if $f_k$ is the random solution to Msat described by $T_k$, then the ratio $r_k = f_k/m_k$ has an expected value $E[r_k] \geqq \frac{1}{2}$. We begin by proving the following lemma.

LEMMA 1. *$a_j = m_j - x_j$ and $\bar{a}_j \geqq \max\{0, m_j - n_j\}, j = 1, \cdots, k$.*

*Proof.* As $u_j$, the optimal column for Msat described by table $T_j$, has $x_j$ 1's, the optimal solution value to Msat described by table $T(u_j)$ is, trivially, $a_j = m_j - x_j$. Also, $x_j \leqq n_j$ (by definition), so that $m_j - x_j \geqq m_j - n_j$. Of the $m_j - x_j$ rows with at least one 1 in $T(u_j)$, at most $n_j - x_j$ can also have 1's in column $\bar{u}_j$ of $T_j$. Hence the Msat problem described by $T(\bar{u}_j)$ has an optimal solution with value $\bar{a}_j$ no smaller than $m_j - x_j - (n_j - x_j) = m_j - n_j$. As $n_j$ can exceed $m_j$, and as the value of the optimal solution to Msat described by $T_j$ is nonnegative, $\bar{a}_j \geqq \max\{0, m_j - n_j\}, j = 1, \cdots, k$. □

THEOREM 1. *$E[r_k] \geqq \frac{1}{2}$ for all $k$.*

*Proof.* We prove the theorem by induction on the number of variables.

*Base case.* $E[r_1] \geqq \frac{1}{2}$.

As each column of $T_1$ is selected with probability $\frac{1}{2}$, the expected value of the random solution is

$$E[r_1] = \frac{1}{2}x_1 + \frac{1}{2}(n_1 - x_1) = \frac{n_1}{2}.$$

As the optimal solution value, corresponding to $u_1$, is $m_1 = x_1 \leqq n_1$, the expected performance ratio is

$$E[r_1] = \frac{(n_1/2)}{x_1} \geq \frac{(n_1/2)}{n_1} = \frac{1}{2}.$$

*Induction hypothesis.* $E[r_l] \geqq \frac{1}{2}$ for all $l \leqq k - 1$.

*Induction step.* To prove $E[r_k] \geqq \frac{1}{2}$.

| Row | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 1 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |

FIG. 3. *Table* $T_2 = T(\bar{u}_3)$ *for Example* 1.

By the induction hypothesis, the random procedure applied to table $T(u_k)(T(\bar{u}_k))$ has an expected solution value no smaller than $\frac{1}{2}a_k(\frac{1}{2}\bar{a}_k)$. Hence the random solution, for table $T_k$, has an expected solution

$$E[f_k] \geqq \frac{1}{2}\left(x_k + \frac{a_k}{2}\right) + \frac{1}{2}\left(n_k - x_k + \frac{\bar{a}_k}{2}\right).$$

As $a_k = m_k - x_k$ by Lemma 1,

$$E[f_k] \geqq \frac{1}{2}\left(x_k + \frac{m_k - x_k}{2}\right) + \frac{1}{2}\left(n_k - x_k + \frac{\bar{a}_k}{2}\right).$$

Also, $\bar{a}_k \geqq \max\{0, m_k - n_k\}$ by Lemma 1. Consider $m_k > n_k$. Then $\bar{a}_k \geqq m_k - n_k > 0$, and the above inequality for $E[f_k]$ simplifies to

$$E[f_k] \geqq \frac{1}{2}\left(x_k + \frac{m_k - x_k}{2}\right) + \frac{1}{2}\left(n_k - x_k + \frac{m_k - n_k}{2}\right),$$

or

$$E[f_k] \geqq \frac{m_k}{2} + \frac{n_k}{4} - \frac{x_k}{4}.$$

The right-hand side attains the least value at $x_k = n_k$, at which value

$$E[f_k] \geqq \frac{m_k}{2},$$

and

$$E[r_k] = \frac{E[f_k]}{m_k} \geqq \frac{1}{2}.$$

Now consider $m_k \leqq n_k$. Then $\bar{a}_k \geqq 0$ $(\geqq m_k - n_k)$, and hence

$$E[f_k] \geqq \frac{1}{2}\left(x_k + \frac{m_k - x_k}{2}\right) + \frac{1}{2}(n_k - x_k).$$

Simplifying,

$$E[f_k] \geqq \frac{m_k}{4} + \frac{n_k}{2} - \frac{x_k}{4}.$$

Since $m_k \leqq n_k$, the right-hand side attains the least value at $x_k = n_k = m_k$, at which value

$$E[f_k] \geqq \frac{m_k}{2},$$

and $E[r_k] = E[f_k]/m_k \geqq \frac{1}{2}$. □

The lower bound on the average value $E[r_k]$ of $r_k$, is tight and is illustrated by the example in Fig. 4. Assume that the data pattern shown in the figure is generated each time; i.e., the data-generating mechanism presents the same pattern with $x$ 1's in column $u_2$ and $(n_2 - x)$ 1's in column $u_1$, where $x$ can range from 1 to $n_2$. The probability of a particular value of $x$ for a problem instance is determined by the distribution of the random variable $x$. The average performance of the random solution is the average of the performance ratio across the four solutions that can be selected, each solution being selected with probability $\frac{1}{4}$. The average performance ratio can be verified to be $\frac{1}{2}$, which is the lower bound on the expected performance ratio for random search.

| Row | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| · | · | · | · | · |
| · | · | · | · | · |
| $x$ | 1 | 0 | 0 | 0 |
| $x+1$ | 0 | 0 | 1 | 0 |
| · | · | · | · | · |
| · | · | · | · | · |
| $n_2$ | 0 | 0 | 1 | 0 |

FIG. 4. *Worst case example for random procedure. $x$ is a discrete random variable ranging from* 1 *to $n_2$.*

**3. The greedy heuristic.** Random search, of course, appears to be a simplistic procedure for solving the problem. A greedy heuristic that selects columns based on the number of 1's they contain is presented next (Johnson [13]), and its worst-case performance and average performance are analyzed. We begin by describing the greedy heuristic.

*Initialization.* Order the columns of $T_k$ so that $n_k$, the number of 1's across $u_k$ and $\bar{u}_k$, is largest among all pairs of columns $u_l$, $\bar{u}_l$, $l = 1, \cdots, k$ (the ordering plays no role in the analysis and is used merely to detect the termination of the algorithm efficiently). If $x_k \geqq n_k - x_k$, select column $u_k$; otherwise, select column $\bar{u}_k$. Eliminate $u_k$ and $\bar{u}_k$, and all rows with a 1 in the chosen column. Note that the resulting table is denoted by $T_{k-1}$. That is,

$$T_{k-1} = \begin{cases} T(u_k), & \text{if column } u_k \text{ is chosen from } T_k, \\ T(\bar{u}_k), & \text{if column } \bar{u}_k \text{ is chosen from } T_k. \end{cases}$$

*Recursion.* Order the columns of $T_j$ and rename the variables $u_1$ through $u_j$ so that $n_j$, the number of 1's across $u_j$ and $\bar{u}_j$, is largest among all pairs of columns $u_l$, $\bar{u}_l$, $l = 1, \cdots, j$. If $x_j \geqq n_j - x_j$, select column $u_j$; otherwise, select column $\bar{u}_j$. Eliminate $u_j$ and $\bar{u}_j$, and all rows with a 1 in the chosen column. Again, note that the resulting table is denoted by $T_{j-1}$. That is,

$$T_{j-1} = \begin{cases} T(u_j), & \text{if column } u_j \text{ is chosen from } T_j, \\ T(\bar{u}_j), & \text{if column } \bar{u}_j \text{ is chosen from } T_j. \end{cases}$$

*Termination.* Stop if $T_j$ contains no 1's, or if $j = 0$.

Note that

$$m_{j-1} = \begin{cases} a_j, & \text{if } T_{j-1} = T(u_j), \\ \bar{a}_j, & \text{if } T_{j-1} = T(\bar{u}_j), \end{cases}$$

denotes the value of the optimal solution to Msat described by $T_{j-1}$, $j = 1, \cdots, k$. Let $f_j$ denote the value of the greedy solution, and let $r_j = f_j/m_j$ denote the performance ratio of the greedy heuristic for Msat described by $T_j$. Theorem 2 shows that the worst-case bound for the greedy heuristic is $\frac{1}{2}$ of the optimal. We also show by an example that this lower bound on the performance of the greedy heuristic is tight. We begin by proving the following lemma.

LEMMA 2. $m_{j-1} \geqq m_j - n_j$, $j = 1, \cdots, k$.

*Proof.* By Lemma 1, $a_j = m_j - x_j$. As $x_j \leqq n_j$ (by definition), $a_j \geqq m_j - n_j$. Also, $\bar{a}_j \geqq \max\{0, m_j - n_j\} \geqq m_j - n_j$ by Lemma 1. As $m_{j-1}$, the value of the optimal solution to Msat described by $T_{j-1}$, is either $a_j$ or $\bar{a}_j$, it follows that $m_{j-1} \geqq m_j - n_j$. □

THEOREM 2. *$r_k \geqq \frac{1}{2}$ for all $k$.*

*Proof.* We prove the theorem by induction on the number of variables.

*Base case.* $r_1 \geqq \frac{1}{2}$.

The single-variable problem is described by table $T_1$ with column $u_1$ containing $x_1$ 1's, and column $\bar{u}_1$ containing $(n_1 - x_1)$ 1's. The greedy heuristic selects the column with more 1's, which also is the optimal column. Thus

$$f_1 = m_1 = \max\{x_1, n_1 - x_1\},$$

and hence

$$r_1 = \frac{f_1}{m_1} = 1 \geqq \frac{1}{2}.$$

*Induction hypothesis.* $r_j \geqq \frac{1}{2}$ for all $j \leqq k-1$.

*Induction step.* To prove $r_k \geqq \frac{1}{2}$.

At the first step, the greedy heuristic chooses $u_k$ or $\bar{u}_k$, whichever has the larger number of 1's. Hence

$$f_k = \max\{x_k, n_k - x_k\} + f_{k-1}.$$

By the induction hypothesis, $r_{k-1} \geqq \frac{1}{2}$, so that

$$f_{k-1} = r_{k-1} m_{k-1} \geqq \frac{1}{2} m_{k-1}.$$

Hence,

$$f_k \geqq \max\{x_k, n_k - x_k\} + \frac{1}{2} m_{k-1}.$$

As the maximum of two numbers is no smaller than their mean, $\max\{x_k, n_k - x_k\} \geqq \frac{1}{2} n_k$. Also, by Lemma 2, $m_{k-1} \geqq m_k - n_k$. Thus,

$$f_k \geqq \frac{1}{2} n_k + \frac{1}{2}(m_k - n_k) = \frac{m_k}{2}.$$

Thus $r_k = f_k / m_k \geqq \frac{1}{2}$. □

The bound specified by Theorem 2 is tight, and is illustrated by the example in Fig. 5. The optimal solution is $m_k = 2^k$, corresponding to columns $u_j$, $j = 1, \cdots, k$. The greedy solution is $f_k = 2^{k-1} + 1$, corresponding to column $\bar{u}_k$. Hence $r_k = f_k / m_k = \frac{1}{2} + \varepsilon$, where $\varepsilon = 1/2^k$. Since $\varepsilon$ can be made to approach 0 arbitrarily closely by increasing $k$, $r_k$ can be made to approach $\frac{1}{2}$ from above arbitrarily closely, giving rise to an asymptotic upper bound of $\frac{1}{2}$ for the worst-case performance of the greedy heuristic. Observe that the worst-case bound for the greedy heuristic equals the average-case bound for the random solution.

We are now ready to prove the lower bound on the average performance of the greedy heuristic. Assume that a probabilistic data-generating mechanism is used to obtain instances of Msat. Specifically, assume that the mechanism generates a larger number of 1's in $u_j$ with probability $p$, and generates a larger number of 1's in $\bar{u}_j$ with probability $(1-p)$, $j = 1, \cdots, k$. Note that we assume that $p$ does not vary with $j$. However, we make no distributional assumptions about the data-generating process. Theorem 3 characterizes the lower bound on the average performance ratio for the greedy heuristic.

THEOREM 3.

$$E[r_k] \geqq \frac{1}{2-p}.$$

| Row | $u_k$ | $\bar{u}_k$ | $u_{k-1}$ | $\bar{u}_{k-1}$ | · | · | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | · | · | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | · | · | 1 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | · | · | 0 | 0 | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $2^{k-2}$ | 0 | 1 | 0 | 1 | · | · | 0 | 0 | 0 | 0 |
| $2^{k-2}+1$ | 0 | 1 | 1 | 0 | · | · | 0 | 0 | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $2^{k-1}$ | 0 | 1 | 1 | 0 | · | · | 0 | 0 | 0 | 0 |
| $2^{k-1}+1$ | 1 | 0 | 0 | 0 | · | · | 0 | 0 | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $2^k$ | 1 | 0 | 0 | 0 | · | · | 0 | 0 | 0 | 0 |
| $2^k+1$ | 0 | 1 | 0 | 1 | · | · | 0 | 1 | 0 | 1 |

FIG. 5. *Worst case example for the greedy heuristic with $n_k = 2^k + 1$ clauses.*

*Proof.* We prove the theorem by induction on the number of variables.

*Base case.* $E[r_1] \geqq 1/(2-p)$.

For a single-variable problem, the optimal column $u_1$ has at least as many 1's as the nonoptimal column $\bar{u}_1$. Therefore the value of the greedy solution equals the number of 1's in the optimal column. Thus, the expected performance ratio of the greedy heuristic is

$$E[r_1] = \frac{E[f_1]}{m_1} = 1 \geqq \frac{1}{2-p} \quad \text{for any } p, \qquad 0 \leqq p \leqq 1.$$

*Induction hypothesis.* $E[r_l] \geqq 1/(2-p)$ for all $l \leqq k-1$.

*Induction step.* To prove $E[r_k] \geqq 1/(2-p)$.

If the greedy heuristic selects column $u_k$ from $T_k$, it guarantees a solution value of at least $x_k$. In addition, table $T_{k-1} = T(u_k)$, generated at the first step, describes an Msat problem for which the expected value of the greedy solution is, by the induction hypothesis, no less than $[1/(2-p)]a_k$. Hence if column $u_k$ is selected at step 1, the expected value of the greedy solution is no less than $x_k + [1/(2-p)]a_k$. By a similar argument, if the greedy heuristic selects $\bar{u}_k$ at step 1, the expected value of its solution is no less than $n_k - x_k + [1/(2-p)]\bar{a}_k$. Now $u_k$ is selected with probability $p$, and $\bar{u}_k$ is selected with probability $(1-p)$. The expected value of the greedy solution is therefore

$$E[f_k] \geqq p\left(x_k + \frac{1}{2-p}a_k\right) + (1-p)\left(n_k - x_k + \frac{1}{2-p}\bar{a}_k\right).$$

Noting that $a_k = m_k - x_k$ by Lemma 1,

$$E[f_k] \geqq p\left(x_k + \frac{1}{2-p}(m_k - x_k)\right) + (1-p)\left(n_k - x_k + \frac{1}{2-p}\bar{a}_k\right).$$

Also, $\bar{a}_k \geqq \max\{0, m_k - n_k\}$ by Lemma 1. Consider $m_k > n_k$. Then $\bar{a}_k \geqq m_k - n_k > 0$, and the above inequality for $E[f_k]$ becomes

$$E[f_k] \geqq p\left(x_k + \frac{1}{2-p}(m_k - x_k)\right) + (1-p)\left(n_k - x_k + \frac{1}{2-p}(m_k - n_k)\right).$$

Simplifying,

$$E[f_k] \geqq p\left(\frac{1-p}{2-p}x_k + \frac{1}{2-p}m_k\right) + (1-p)\left(n_k - x_k + \frac{1}{2-p}(m_k - n_k)\right).$$

Noting that $x_k \geqq (n_k)/2$ if column $u_k$ is chosen and $n_k - x_k \geqq (n_k)/2$ if column $\bar{u}_k$ is chosen, we get

$$E[f_k] \geqq p\left(\frac{1-p}{2-p}\frac{n_k}{2} + \frac{1}{2-p}m_k\right) + (1-p)\left(\frac{n_k}{2} + \frac{1}{2-p}(m_k - n_k)\right)$$

which implies that

$$E[f_k] \geqq \frac{m_k}{2-p}.$$

Hence

$$E[r_k] = \frac{E[f_k]}{m_k} \geqq \frac{1}{2-p}.$$

Now consider $m_k \leqq n_k$. Then $\bar{a}_k \geqq 0$ $(\geqq m_k - n_k)$, which implies

$$E[f_k] \geqq p\left(\frac{1-p}{2-p}\frac{n_k}{2} + \frac{1}{2-p}m_k\right) + (1-p)\left(\frac{n_k}{2}\right).$$

Simplifying,

$$E[f_k] \geqq \frac{pm_k + (1-p)n_k}{2-p}.$$

As $m_k \leqq n_k$, the right side of the above expression has a minimum at $n_k = m_k$. Hence

$$E[f_k] \geqq \frac{pm_k + (1-p)m_k}{2-p} = \frac{m_k}{2-p}.$$

Thus, $E[r_k] = (E[f_k])/m_k \geqq 1/(2-p)$. □

The lower bound obtained in Theorem 3 is tight. To illustrate, consider the following example involving $k$ variables and $n_k = 2^s - 1$ rows, where $s > k$. Generate the data as follows. For $j = 1, \cdots, k$, set

$$\begin{cases} t_{i,k-j+1} = 0, \bar{t}_{i,k-j+1} = 1, & \text{if } i = 1, \cdots, 2^{s-j} - 1 \\ t_{i,k-j+1} = 1, \bar{t}_{i,k-j+1} = 0, & \text{if } i = 2^{s-j} + 1, \cdots, 2^{s-j+1} - 1 \\ t_{i,k-j+1} = 1, \bar{t}_{i,k-j+1} = 0 \text{ with probability } p, & \text{if } i = 2^{s-j} \\ t_{i,k-j+1} = 0, \bar{t}_{i,k-j+1} = 1 \text{ with probability } 1-p, & \text{if } i = 2^{s-j}. \end{cases}$$

Generate 0's in all remaining cells.

Figure 6 is an example of the data generated for $k = 2$, $n_2 = 2^3 - 1 = 7$. The data generated in this manner for arbitrary $k$ and $n_k = 2^{k+1} - 1$ is shown in Fig. 7.

Since only one 1 is generated probabilistically in column $u_j$ or $\bar{u}_j$, the probability of the greedy heuristic choosing $u_j$ is $p$, and the probability of choosing $\bar{u}_j$ is $(1-p)$, $j = 1, \cdots, k$. Note that each row has a 1 in either column $u_k$ or column $\bar{u}_k$. The value of

| Row | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | $x_1$ | $y_1$ |
| 3 | 0 | 1 | 1 | 0 |
| 4 | $x_2$ | $y_2$ | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 |

FIG. 6. *Data generated for* $k = 2$, $n_2 = 2^3 - 1$. $x_i\ y_i$ *equals* 1 0 *with probability* $p$, 0 1 *with probability* $1 - p$, *for* $i = 1, 2, \cdots, k$.

| Row | $u_k$ | $\bar{u}_k$ | $u_{k-1}$ | $\bar{u}_{k-1}$ | $u_{k-2}$ | $\bar{u}_{k-2}$ | · | · | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | · | · | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 | 1 | · | · | $x_1$ | $y_1$ |
| 3 | 0 | 1 | 0 | 1 | 0 | 1 | · | · | 1 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 | · | · | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $\frac{n_k+1}{8} - 1$ | 0 | 1 | 0 | 1 | 0 | 1 | · | · | 0 | 0 |
| $\frac{n_k+1}{8}$ | 0 | 1 | 0 | 1 | $x_{k-2}$ | $y_{k-2}$ | · | · | 0 | 0 |
| $\frac{n_k+1}{8} + 1$ | 0 | 1 | 0 | 1 | 1 | 0 | · | · | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $\frac{n_k+1}{4} - 1$ | 0 | 1 | 0 | 1 | 1 | 0 | · | · | 0 | 0 |
| $\frac{n_k+1}{4}$ | 0 | 1 | $x_{k-1}$ | $y_{k-1}$ | 0 | 0 | · | · | 0 | 0 |
| $\frac{n_k+1}{4} + 1$ | 0 | 1 | 1 | 0 | 0 | 0 | · | · | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $\frac{n_k+1}{2} - 1$ | 0 | 1 | 1 | 0 | 0 | 0 | · | · | 0 | 0 |
| $\frac{n_k+1}{2}$ | $x_k$ | $y_k$ | 0 | 0 | 0 | 0 | · | · | 0 | 0 |
| $\frac{n_k+1}{2} + 1$ | 1 | 0 | 0 | 0 | 0 | 0 | · | · | 0 | 0 |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| · | · | · | · | · | · | · | · | · | · | · |
| $n_k$ | 1 | 0 | 0 | 0 | 0 | 0 | · | · | 0 | 0 |

FIG. 7. *Data generated for arbitrary* $k$ *and* $n_k = 2^{k+1} - 1$. $x_i\ y_i$ *equals* 1 0 *with probability* $p$, 0 1 *with probability* $1 - p$, *for* $i = 1, 2, \cdots, k$.

the optimal solution to Msat described by $T_k$ is $m_k \geqq n_k - k$, because the collection of columns $u_j, j = 1, \cdots, k$, has at least $n_k - k$ nonoverlapping 1's. The expected performance of the greedy heuristic is

$$\begin{aligned} E[f_k] &= p\frac{n_k+1}{2} + (1-p)\frac{n_k+1}{2} + p\left(p\frac{n_k+1}{4} + (1-p)\frac{n_k+1}{4}\right) \\ &+ p^2\left(p\frac{n_k+1}{8} + (1-p)\frac{n_k+1}{8}\right) + \cdots \\ &+ p^{k-1}\left(p\frac{n_k+1}{2^k} + (1-p)\frac{n_k+1}{2^k}\right) \\ &= \frac{n_k+1}{2}\left(1 + \frac{p}{2} + \left(\frac{p}{2}\right)^2 + \cdots + \left(\frac{p}{2}\right)^{k-1}\right). \end{aligned}$$

Since

$$\left(1 + \frac{p}{2} + \left(\frac{p}{2}\right)^2 + \cdots + \left(\frac{p}{2}\right)^{k-1}\right) = \frac{2}{2-p}\left(1 - \left(\frac{p}{2}\right)^k\right),$$

and

$$E[r_k] = \frac{E[f_k]}{m_k},$$

we get

$$E[r_k] \leqq \frac{n_k+1}{2m_k}\frac{2}{2-p}\left(1 - \left(\frac{p}{2}\right)^k\right).$$

As $m_k \geqq n_k - k$,

$$E[r_k] \leqq \frac{n_k+1}{n_k-k}\frac{1}{2-p}\left(1 - \left(\frac{p}{2}\right)^k\right).$$

Let $\varepsilon_1 = (k+1)/(n_k - k)$ and $\varepsilon_2 = (p/2)^k$. Since $n_k > 2^k - 1$ and $p \geqq 0$, it follows that $\varepsilon_1 > 0$ and $\varepsilon_2 \geqq 0$ for all $k \geqq 1$. $E[r_k]$ may then be written as

$$E[r_k] \leqq (1+\varepsilon_1)(1-\varepsilon_2)\frac{1}{2-p}$$

implying that

$$E[r_k] \leqq (1+\varepsilon_1)\frac{1}{2-p}.$$

Since $n_k > 2^k - 1$, $n_k$ grows exponentially faster than $k$. Consequently, $\varepsilon_1 = (k+1)/(n_k - k)$ approaches 0 for large $k$. Hence the upper bound on $E[r_k]$ can be made arbitrarily close to $1/(2-p)$. As it is also a lower bound on $E[r_k]$ by Theorem 3, $1/(2-p)$ is a tight lower bound on $E[r_k]$.

How small can $p$, and hence $1/(2-p)$, be? For the data-generating mechanism described above, $p$ can be arbitrarily small. The lower bound on the average performance ratio for the greedy heuristic then approaches $\frac{1}{2}$, the same as the lower bound on the average performance ratio for the random procedure. However, the data-generating mechanism described above is "perverse." Other mechanisms can possibly guarantee higher minimum values for $p$, and hence a higher minimum performance ratio for the greedy heuristic. One mechanism, similar to that used in Goldberg, Purdom, and Brown

[11], is as follows: for all $j = 1, \cdots, k$, generate

$$\begin{cases} t_{ij}=1, \bar{t}_{ij}=0, & \text{with probability } q, \\ t_{ij}=0, \bar{t}_{ij}=1, & \text{with probability } q, \\ t_{ij}=0, \bar{t}_{ij}=0, & \text{with probability } 1-2q. \end{cases}$$

The choice of $q$ is arbitrary, and as in many simulations may be based on random sampling from a parametric distribution. In this case, the probability that $u_j$ has a larger number of 1's than $\bar{u}_j$ is $\frac{1}{2}$, $j = 1, \cdots, k$, and hence $E[r_k] \geqq \frac{2}{3}$.

Is there a way to improve the lower bound on $E[r_k]$ for the greedy heuristic? So long as $p$ is governed by "nature" (i.e., by a data-generating mechanism which the algorithm cannot control), there appears to be no way. But there is no reason why the choice of $p$ should not be made a part of the heuristic. For instance, we may introduce probabilistic choice at each step of the greedy heuristic so that, whatever $p$ is, the heuristic selects a solution with a probability *it chooses*. The perversity of a data-generating mechanism may then be superseded by the heuristic. We pursue this approach below by describing a probabilistic version of the greedy heuristic, which we call the *probabilistic greedy heuristic*.

**4. The probabilistic greedy algorithm.** Like the greedy heuristic, the probabilistic greedy heuristic selects at step $j$ column $u_j$ or $\bar{u}_j$ from table $T_j$, $j = 1, \cdots, k$. However, each column is selected with probability proportional to the number of 1's it contains. That is, $u_j$ is chosen with probability $p = x_j/n_j$, and $\bar{u}_j$ is chosen with probability $1 - p = (n_j - x_j)/n_j$. We describe the heuristic more formally below.

*Initialization.* Order the columns of $T_k$ so that $n_k$, the number of 1's across $u_k$ and $\bar{u}_k$, is largest among all pairs of columns $u_l$, $\bar{u}_l$, $l = 1, \cdots, k$. Select column $u_k$ with probability $p = x_k/n_k$, and select column $\bar{u}_k$ with probability $1 - p = (n_k - x_k)/n_k$. Eliminate $u_k$ and $\bar{u}_k$, and all rows with a 1 in the chosen column, to obtain table $T_{k-1}$, where, as before,

$$T_{k-1} = \begin{cases} T(u_k), & \text{if column } u_k \text{ is chosen from } T_k, \\ T(\bar{u}_k), & \text{if column } \bar{u}_k \text{ is chosen from } T_k. \end{cases}$$

*Recursion.* Order the columns of $T_j$ so that $n_j$, the number of 1's across $u_j$ and $\bar{u}_j$, is largest among all pairs of columns $u_l$, $\bar{u}_l$, $l = 1, \cdots, j$. Select column $u_j$ with probability $p = x_j/n_j$, and select column $\bar{u}_j$ with probability $1 - p = (n_j - x_j)/n_j$. Eliminate $u_j$ and $\bar{u}_j$, and all rows with a 1 in the chosen column, to obtain table $T_{j-1}$, where, as before,

$$T_{j-1} = \begin{cases} T(u_j), & \text{if column } u_j \text{ is chosen from } T_j, \\ T(\bar{u}_j), & \text{if column } \bar{u}_j \text{ is chosen from } T_j. \end{cases}$$

*Termination.* Stop if $T_j$ contains no 1's, or if $j = 0$.

The probabilistic greedy heuristic forces a trade-off between the probability of selecting the optimal solution and the value of the nonoptimal solution it identifies. We illustrate the trade-off below for the Msat problem described by $T_k$. Assume that at each of the first $k - 1$ steps the probabilistic greedy heuristic chooses the optimal column. At step $k$, the probabilistic greedy heuristic chooses column $u_1$ with probability $p = x_1/n_1$, and column $\bar{u}_1$ with probability $1 - p = (n_1 - x_1)/n_1$. Hence the expected performance ratio is

$$E[r_k] = \frac{x_1}{n_1}\frac{x_1 + (m_1 - x_1)}{m_1} + \frac{n_1 - x_1}{n_1}\frac{(n_1 - x_1) + (m_1 - x_1)}{m_1}$$

where $m_1 - x_1$ is the number of clauses satisfied by the optimal columns selected by the greedy heuristic in steps 1 to $k-1$, and hence $x_1 + m_1 - x_1$ is the value of the greedy solution if column $u_1$ is selected and $n_1 - x_1 + m_1 - x_1$ is the value of the greedy solution if column $\bar{u}_1$ is selected. The trade-off can be seen in the expression for $E[r_k]$. The probability of selecting the optimal column $u_1$ decreases as $x_1$ decreases. However, as $x_1$ decreases, the value of the nonoptimal solution $n_1 - x_1 + m_1 - x_1 = n_1 + m_1 - 2x_1$ increases. The lower bound on $E[r_k]$ is obtained by choosing $x_1$ so that $E[r_k]$ has its smallest value. It can be verified that $E[r_k]$ is minimized by setting $x_1 = 3n_1/4$, at which value, $E[r_k] = 1 - n_1/8m_1$. Hence $m_1 \geqq x_1 = 3n_1/4$. Thus

$$\min E[r_k] = 1 - \frac{n_1}{8m_1} \geqq 1 - \frac{n_1}{8(3n_1/4)} = \frac{5}{6}.$$

That is, the lower bound on the expected performance ratio is $\frac{5}{6}$ when the first $k-1$ columns selected by the greedy heuristic are optimal. As described below, the trade-off between the probability of selecting the optimal solution and the value of the nonoptimal solution occurs in general for the probabilistic greedy heuristic.

THEOREM 4. *$E[r_k] \geqq \frac{2}{3}$ for all $k$.*

*Proof.* We prove the theorem by induction on the number of variables.

*Base case.* $E[r_1] \geqq \frac{2}{3}$.

For the single-variable problem, the optimal solution to Msat described by $T_1$ is $m_1 = x_1$, and corresponds to column $u_1$ of $T_1$ as per our assumption. As the probabilistic greedy heuristic selects $u_1$ with probability $p = x_1/n_1$, and selects $\bar{u}_1$ with probability $1 - p = (n_1 - x_1)/n_1$, the expected value of its solution is

$$E[f_1] = px_1 + (1-p)(n_1 - x_1),$$

and the expected performance ratio of the heuristic is

$$E[r_1] = \frac{E[f_1]}{m_1} = \frac{[(x_1/n_1)]x_1 + [(n_1 - x_1)/n_1](n_1 - x_1)}{x_1}.$$

Given $n_1$, the lower bound on $E[r_1]$ is obtained by minimizing the above expression with respect to $x_1$, which can be verified to occur at $x_1 = n_1/\sqrt{2}$. Substituting this value of $x_1$ in $E[r_1]$ and simplifying yields

$$E[r_1] \geqq 2\sqrt{2} - 2 \geqq \frac{2}{3}.$$

*Induction hypothesis.* $E[r_l] \geqq \frac{2}{3}$ for all $l \leqq k-1$.

*Induction step.* $E[r_k] \geqq \frac{2}{3}$ for all $k$.

If the probabilistic greedy heuristic selects column $u_k$ from $T_k$, it guarantees a solution value of at least $x_k$. In addition, $T_{k-1} = T(u_k)$, generated at the first step, describes an Msat problem for which the expected value of the heuristic solution is, by the induction hypothesis, no less than $\frac{2}{3}a_k$. Hence if column $u_k$ is selected at step 1, the expected value of the heuristic solution is no less than $x_k + \frac{2}{3}a_k$. By a similar argument, if the greedy heuristic selects $\bar{u}_k$ at step 1, the expected value of its solution is no less than $n_k - x_k + \frac{2}{3}\bar{a}_k$. Now $u_k$ is selected with probability $p = x_k/n_k$, and $\bar{u}_k$ is selected with probability $1 - p = (n_k - x_k)/n_k$. The expected value of the heuristic solution is therefore

$$E[f_k] \geqq \frac{x_k}{n_k}\left(x_k + \frac{2}{3}a_k\right) + \frac{n_k - x_k}{n_k}\left(n_k - x_k + \frac{2}{3}\bar{a}_k\right).$$

As $a_k = m_k - x_k$ by Lemma 1,

$$E[f_k] \geqq \frac{x_k}{n_k}\left(x_k + \frac{2}{3}(m_k - x_k)\right) + \frac{n_k - x_k}{n_k}\left(n_k - x_k + \frac{2}{3}\bar{a}_k\right).$$

Also, $\bar{a}_k \geqq \max\{0, m_k - n_k\}$ by Lemma 1. Consider $m_k > n_k$. Then $\bar{a}_k \geqq m_k - n_k > 0$, and the above inequality for $E[f_k]$ becomes

$$E[f_k] \geqq \frac{x_k}{n_k}\left(\frac{2m_k}{3}+\frac{x_k}{3}\right)+\frac{(n_k-x_k)^2}{n_k}+\frac{2(n_k-x_k)}{3n_k}(m_k-n_k).$$

Simplifying,

$$E[f_k] \geqq \frac{(x_k)^2}{3n_k}+\frac{2m_kx_k}{3n_k}+\frac{(n_k-x_k)^2}{n_k}+\frac{2(n_k-x_k)}{3n_k}(m_k-n_k).$$

The right side of the above expression can be verified to obtain its minimum value when $x_k = n_k/2$, at which value of $x_k$,

$$E[f_k] \geqq \frac{2m_k}{3},$$

and hence

$$E[r_k] = \frac{E[f_k]}{m_k} \geqq \frac{2}{3}.$$

Now consider $m_k \leqq n_k$. Then $\bar{a}_k \geqq 0$ ($\geqq m_k - n_k$), and hence

$$E[f_k] \geqq \frac{x_k}{n_k}\left(x_k+\frac{2}{3}(m_k-x_k)\right)+\frac{n_k-x_k}{n_k}(n_k-x_k).$$

Simplifying

$$E[f_k] \geqq \frac{(x_k)^2}{3n_k}+\frac{2m_kx_k}{3n_k}+\frac{(n_k-x_k)^2}{n_k}.$$

The right side of the above expression can be verified to obtain its minimum value when $x_k = (3n_k - m_k)/4$, at which value of $x_k$

$$E[f_k] \geqq \frac{n_k}{4}+\frac{m_k}{2}-\frac{m_k^2}{12n_k}.$$

The right side of the above expression takes its smallest value when $n_k = m_k$, for which

$$E[f_k] \geqq \frac{m_k}{4}+\frac{m_k}{2}-\frac{m_k}{12}=\frac{2}{3}m_k.$$

Therefore $E[r_k] = E[f_k]/m_k \geqq \frac{2}{3}$. □

It can be verified that for the data in Fig. 8, the expected performance of the probabilistic greedy heuristic is

$$E[f_k] = 2\left(\frac{1}{2}\right)\frac{n_k}{2}+2\left(\frac{1}{2}\right)^2\frac{n_k}{2^2}+\cdots+2\left(\frac{1}{2}\right)^k\frac{n_k}{2^k}+\left(\frac{1}{2}\right)^k\frac{n_k}{2^k}$$
$$=\frac{2n_k}{3}+\frac{n_k}{3}\left(\frac{1}{4^k}\right).$$

Noting that $m_k = n_k$, the expected performance ratio equals $E[r_k] = \frac{2}{3} + \varepsilon$, where $\varepsilon = \frac{1}{3}(1/4^k)$. Since $\varepsilon$ can be made to approach 0 arbitrarily closely by increasing $k$, $E[r_k]$ can be made to approach $\frac{2}{3}$ from above arbitrarily closely. Since the asymptotic upper bound for the probabilistic greedy heuristic is $\frac{2}{3}$, the lower bound of $\frac{2}{3}$ specified by Theorem 4 is tight.

As the data-generating mechanism plays no role in determining the lower bound of the performance ratio for the probabilistic greedy heuristic, the bound obtained by

| Row | $u_k$ | $\bar{u}_k$ | $u_{k-1}$ | $\bar{u}_{k-1}$ | · | · | $u_2$ | $\bar{u}_2$ | $u_1$ | $\bar{u}_1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | · | · | 0 | 1 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | · | · | 1 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | · | · | 0 | 0 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $2^{k-2}$ | 0 | 1 | 0 | 1 | · | · | 0 | 0 | 0 | 0 |
| $2^{k-2}+1$ | 0 | 1 | 1 | 0 | · | · | 0 | 0 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $2^{k-1}$ | 0 | 1 | 1 | 0 | · | · | 0 | 0 | 0 | 0 |
| $2^{k-1}+1$ | 1 | 0 | 0 | 0 | · | · | 0 | 0 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $2^k$ | 1 | 0 | 0 | 0 | · | · | 0 | 0 | 0 | 0 |

FIG. 8. *Worst case example for the probabilistic greedy heuristic with $n_k = 2^k$ clauses.*

Theorem 4 also holds if the *same* problem is sampled repeatedly; i.e., if the probabilistic greedy heuristic is implemented multiple times to solve the same problem, the average heuristic solution will be no smaller than $\frac{2}{3}$ of the optimal. Of course, in this case the solution with the largest value is of greater interest than the average value of the solutions across trials. For a large number of trials, the distribution of the largest value of the probabilistic greedy solution corresponds to the extreme value distribution for the largest among a sample of $n$ observations. Since the largest value of the probabilistic greedy solution is bounded from above by the value of the optimal, the distribution in this case is characterized by the limited-value distribution (Gumbel [12]), which corresponds to the type-three distribution in the Fisher and Tippett characterization of extreme-value distributions (Fisher and Tippett [8]). Thus, regardless of the data-generating distribution, the asymptotic cumulative distribution function of the largest value of the probabilistic greedy solution is

$$H[z] = \exp\left(-\left(\frac{m_k - z}{m_k - v}\right)^w\right),$$

with corresponding density

$$h(z) = \frac{w}{m_k - v}\left(\frac{m_k - z}{m_k - v}\right)^{w-1} H(z),$$

where $z$ is the largest value of the probabilistic greedy solution across trials, $H(v) = 1/e = 0.36788$, and $w > 0$ is the shape parameter of the distribution (see, e.g., Gumbel [12, pp. 164–165; p. 275]).

**5. Conclusion.** Two aspects of the probabilistic greedy heuristic should perhaps be mentioned. First, it guarantees an average solution value of no less than $\frac{2}{3}$ of the optimal value regardless of the distribution of data. Second, the trade-off it forces between the probability of selecting the optimal solution and the value of the nonoptimal solution is a feature that is not evidently observed in other heuristics. Indeed, it is this feature of the heuristic that ensures that its average performance is never too bad. In contrast, while the greedy heuristic can do well, its ability to do so depends on the value of $p$. For independent observations from parametric distributions with $p = \frac{1}{2}$, it does as well, on average, as the probabilistic greedy heuristic. But for perverse distributions, the greedy heuristic on average can do as poorly as random search. On the other hand, for an "unintelligent" procedure, the random search does quite well to ensure an average solution of no less than $\frac{1}{2}$ of the optimal, regardless of the data-generating distribution. It remains an open question whether relaxing the independence assumption, or assuming specific distributions, strengthens the bounds on the average performance for the greedy heuristic. It may also be possible to strengthen the average bound for the greedy heuristic with restricting assumptions on problem instances, such as when the set of clauses are 2-satisfiable (Lieberherr and Specker [16]), or when each clause contains at least $l$ variables, $1 \leqq l \leqq k$ (Johnson [13]).

**Acknowledgments.** The authors thank Pavol Hell, Tiko Kameda, Art Liestman, Joe Peters, and Art Warburton for numerous helpful suggestions. Special thanks are due to Tiko Kameda and Joe Peters for carefully reading the paper and giving detailed comments that have helped improve the presentation.

## REFERENCES

[1] J. BENTLEY, D. S. JOHNSON, F. T. LEIGHTON, AND L. A. MCGEOCH, *Some unexpected expected behavior results for bin packing*, Proc. 16th ACM Sympos. Theory of Computing, 1984, pp. 279–288.

[2] O. J. BOXMA, *A probabilistic analysis of multiprocessing list scheduling: the erlang case*, Stochastic Models, 1 (1985), pp. 209–220.

[3] J. L. BRUNO AND P. J. DOWNEY, *Probabilistic bounds for dual bin packing*, Acta Inform., 22 (1985), pp. 333–345.

[4] E. G. COFFMAN AND F. T. LEIGHTON, *A provably efficient algorithm for dynamic storage allocation*, Proc. 18th ACM Sympos. Theory of Computing, 1986, pp. 77–90.

[5] E. G. COFFMAN, G. S. LUEKER, AND A. H. G. RINNOOY KAN, *Asymptotic methods in the probabilistic analysis of sequencing and packing heuristics*, Management Sci., 3 (1988), pp. 266–290.

[6] J. CSIRIK, J. B. G. FRENK, A. FRIEZE, G. GALAMBOS, AND A. H. G. RINNOOY KAN, *A probabilistic analysis of the next fit decreasing bin packing heuristic*, Oper. Res. Lett., 5 (1986), pp. 233–236.

[7] M. DAVIS, G. LOGEMANN, AND D. LOVELAND, *A machine program for theorem proving*, Comm. ACM, 5 (1962), pp. 394–397.

[8] R. A. FISHER AND L. H. C. TIPPETT, *Limiting forms of the frequency distribution of the largest or smallest member of a sample*, Proc. Cambridge Phil. Soc., 24 (1928), pp. 180–190.

[9] J. FRANCO AND M. PAULL, *Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem*, Discrete Applied Math., 5 (1983), pp. 77–87.

[10] J. B. G. FRENK AND A. H. G. RINNOOY KAN, *The asymptotic optimality of the LPT rule*, Math. Oper. Res., 12 (1987), pp. 241–254.

[11] A. GOLDBERG, P. PURDOM, AND C. BROWN, *Average time analysis of simplified Davis–Putnam procedures*, Inform. Process. Lett., 15 (1982), pp. 72–75.

[12] E. J. GUMBEL, *Statistics of Extremes*, Columbia University Press, New York, 1958.

[13] D. S. JOHNSON, *Approximation algorithms for combinatorial problems*, J. Comput. System Sci., 9 (1974), pp. 256–278.

[14] R. M. KARP, M. LUBY, AND A. MARCHETTI-SPACCAMELA, *A probabilistic analysis of multidimensional bin packing problems*, Proc. 16th ACM Sympos. Theory of Computing, 1984, pp. 289–298.

[15] F. T. LEIGHTON AND P. W. SHOR, *Tight bounds for minimax grid matching with applications to the average-case analysis of algorithms,* Proc. 18th ACM Sympos. Theory of Computing, 1986, pp. 91–103.

[16] K. J. LIEBERHERR AND E. SPECKER, *Complexity of partial satisfaction*, J. Assoc. Comput. Mach., 28 (1981), pp. 411–421.

[17] P. W. SHOR, *The average-case analysis of some on-line algorithms for bin packing*, Combinatorica, 6 (1986), pp. 179–200.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 524-529, November 1989

# PARTITIONS OF $A^\omega$*

H. LEFMANN† AND B. VOIGT‡

**Abstract.** A canonizing Ramsey type theorem for Baire mappings $\Delta : A^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space is established.

**Key words.** Ramsey, Baire sets, sequence space

**AMS(MOS) subject classifications.** primary 05A99; secondary 54H05

**1. Introduction.** Let $A$ be a finite set. In this paper we study typical patterns of Baire mappings $\Delta : A^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space. The set $A^\omega$ naturally becomes a topological space by taking the Tychonoff product topology, where $A$ bears the discrete topology.

Moreover, $A^\omega$ is a metric space, where the *Baire metric* is defined by $d((a_n)_{n<\omega}, (b_n)_{n<\omega}) = 1/(k+1)$ if and only if $a_k \neq b_k$, but $a_i = b_i$ for all $i < k$.

For $|A| = 2$ this is the Cantor Discontinuum. As $\{0, 1\}$-sequences can be interpreted as subsets of $\omega$, the set $\{0, 1\}^\omega$ can also be viewed as the power set of $\omega$.

In [PSV86] it is shown that for every Baire mapping $\Delta : \{0, 1\}^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space, there exist subsets $A \subseteq B \subseteq \omega$ such that $B \backslash A$ is infinite and such that $\Delta \restriction \{C \subseteq \omega \mid A \subseteq C \subseteq B\}$ either is a constant or a one-to-one mapping. The question is raised as to whether an analogous result may be established also for $A^\omega$ with $|A| \geqq 3$. We show that this is not quite the case. For some general background see, e.g., [PV85].

**2. Parameter words.** Unless stated otherwise let $A$ be a finite set with $A \cap \{\lambda_0, \lambda_1, \cdots\} = \emptyset$. By $[A]_{one}\binom{\omega}{\omega}$ we denote the set of mappings (words) $F : \omega \to A \cup \{\lambda_i \mid i < \omega\}$ such that $|F^{-1}(\lambda_i)| = 1$ for every $i < \omega$. So each *parameter* $\lambda_i$ occurs exactly once in $F$. For $F \in [A]_{one}\binom{\omega}{\omega}$ and $g \in A^\omega$, we denote by $F \cdot g \in A^\omega$ the insertion of $g$ into the parameters of $F$, i.e.,

$$\begin{aligned}(F\cdot g)(i) &= F(i) \text{ if } F(i) \in A\\ &= g(j) \text{ if } F(i) = \lambda_j,\end{aligned}$$

where $g = (g(0), g(1), \cdots)$.

We also consider *ascending* parameter words, viz., we let $[A]_{asc}\binom{\omega}{\omega}$ denote the set of mappings (words) $F : \omega \to A \cup \{\lambda_i \mid i < \omega\}$ such that $F^{-1}(\lambda_i) \neq \emptyset$ for every $i < \omega$ and, moreover, $\max F^{-1}(\lambda_i) < \min F^{-1}(\lambda_j)$ for all $i < j < \omega$.

For $F \in [A]_{asc}\binom{\omega}{\omega}$ and $g \in A^\omega$ the insertion $F \cdot g \in A^\omega$ is defined as above.

The result from [PSV86] alluded to above can now be formulated as follows:

THEOREM A [PSV86]. *For every Baire mapping $\Delta : \{0, 1\}^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space, there exists an $F \in [\{0, 1\}]_{one}\binom{\omega}{\omega}$ such that $\Delta \restriction F \cdot \{0, 1\}^\omega$ either is a constant or a one-to-one mapping.*

The notion of a *Baire mapping* is explained in the next section. For the moment it may suffice to note that every continuous, respectively, Borel, mapping is also Baire.

* Received by the editors September 19, 1988; accepted for publication November 28, 1988.

† Fakultat fur Mathematik, Universitat Bielefeld, 4800 Bielefeld, Federal Republic of Germany.

‡ Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 5300 Bonn 1, Federal Republic of Germany.

From [PV83] it is well known that with respect to larger sets $A$, i.e., for $|A| \geqq 3$, we cannot expect an $F$ such that $\Delta \upharpoonright F \cdot A^\omega$ is just constant or one-to-one. Additional patterns occur. But these are determined from equivalence relations on the set $A$.

Let $\equiv$ be an equivalence relation on $A$. This induces an equivalence relation on $A^\omega$, by abuse of language, the induced relation is also denoted by $\equiv$, viz., for $g, h \in A^\omega$, let $g \equiv h$ if and only if $g(i) \equiv h(i)$ for all $i < \omega$. Let us call a mapping $\Delta : A^\omega \to \mathscr{Y}$ *canonical* if there exists an equivalence relation $\equiv$ on $A$ such that for all $g, h \in A^\omega$ it follows that

$$\Delta(g) = \Delta(h) \text{ if and only if } g \equiv h.$$

Observe that if $\Delta : A^\omega \to \mathscr{Y}$ is canonical, then for every $F \in [A]_{asc}\binom{\omega}{\omega}$ the restriction $\Delta \upharpoonright F \cdot A^\omega$ is still canonical with respect to the same equivalence relation $\equiv$. In this sense canonical mappings are hereditary, providing typical patterns of continuous mappings $\Delta : A^\omega \to \mathscr{Y}$. The same hereditary patterns also occur with respect to more general concepts of parameter words (cf. [PV85]).

Note that Theorem A may be restated by saying that for every Baire mapping $\Delta : \{0, 1\}^\omega \to \mathscr{Y}$ there exists an $F \in [\{0, 1\}]_{one}\binom{\omega}{\omega}$ such that the restriction $\Delta \upharpoonright F \cdot \{0, 1\}^\omega$ is a canonical mapping. In [PSV86] it has been asked whether this statement also holds for larger sets $A$. We now show that this is not the case.

THEOREM B. *Let $A$ be a finite set with $|A| \geqq 3$. Then there exist continuous mappings $\Delta : A^\omega \to A^\omega$ such that for each $F \in [A]_{one}\binom{\omega}{\omega}$ the restriction $\Delta \upharpoonright F \cdot A^\omega$ is not a canonical mapping.*

However, by allowing a bit more structure, a positive result may be obtained, as shown in Theorem C.

THEOREM C. *Let $A$ be a finite set. For every Baire mapping $\Delta : A^\omega \to \mathscr{Y}$, where $\mathscr{Y}$ is a metric space, there exists an $F \in [A]_{asc}\binom{\omega}{\omega}$ such that the restriction $\Delta \upharpoonright F \cdot A^\omega$ is a canonical mapping.*

In this paper we prove Theorems B and C. On the way we also obtain a short proof of Theorem A.

**3. Topological prerequisites.** In this section we present some topological prerequisites. To keep this paper self-contained we give short proofs of some facts from topology which are perhaps not widely known. Some of these results are valid in a more general setting, e.g., in Lemma 2 up to Lemma 4, the space $A^\omega$ may be replaced by an arbitrary Polish space, and the same proofs still work. For even more general results cf. [EFK79]. The space $A^\omega$ is a complete separable metric space, i.e., a *Polish space*. A set $\mathscr{B} \subseteq A^\omega$ is a *Baire set* (has the property of Baire) if $\mathscr{B}$ may be written as a symmetric difference of an open set $\mathscr{O}$ and a meager set $\mathscr{M}$, i.e., $\mathscr{B} = \mathscr{O} \setminus \mathscr{M} \cup \mathscr{M} \setminus \mathscr{O}$. A set $\mathscr{M}$ is *meager* if it belongs to the complement of a countable intersection of dense open sets.

Moran and Strauss [MS80] have shown that meager sets are Ramsey null:

LEMMA 1. [MS80] *For every meager set $\mathscr{M} \subseteq A^\omega$, there exists an $F \in [A]_{one}\binom{\omega}{\omega}$ such that $(F \cdot A^\omega) \cap \mathscr{M} = \varnothing$.*

A mapping $\Delta : A^\omega \to \mathscr{Y}$, where $\mathscr{Y}$ is a metric space, is a *Baire mapping* if for every open set $\mathscr{O} \subseteq \mathscr{Y}$ its preimage $\Delta^{-1}(\mathscr{O})$ is a Baire set. Baire mappings are very close to continuous mappings. A first step in this direction is due to Kuratowski [Kur30].

LEMMA 2. *For every Baire mapping $\Delta : A^\omega \to \mathscr{Y}$, where $\mathscr{Y}$ is a separable metric space, there exists a meager set $\mathscr{M} \subseteq A^\omega$ such that the restriction $\Delta \upharpoonright A^\omega \setminus \mathscr{M}$ is a continuous mapping.*

*Proof.* Let $(\mathscr{O}_n)_{n<\omega}$ be a basis for $\mathscr{Y}$; this exists because $\mathscr{Y}$ is separable. As $\Delta$ is a Baire mapping each preimage $\Delta^{-1}(\mathscr{O}_n)$ is a Baire set in $A^\omega$, so it may be written as

$\Delta^{-1}(\mathcal{O}_n) = \mathcal{B}_n \backslash \mathcal{M}_n \cup \mathcal{M}_n \backslash \mathcal{B}_n$, where $\mathcal{B}_n$ is open and $\mathcal{M}_n$ is meager. Then $\mathcal{M} = \bigcup_{n<\omega} \mathcal{M}_n$ has the desired properties. $\square$

Lemma 2 may be further extended by dismissing the separability of the metric space $\mathcal{Y}$. Here we use an auxiliary result which is due to Prikry and Solovay [PS78] and to Bukovsky [Bu79].

LEMMA 3. *Let $(\mathcal{M}_s)_{s\in S}$ be a family of mutually disjoint meager sets in $A^\omega$ such that for every $T \subseteq S$ the union $\bigcup_{s\in T}\mathcal{M}_s$ is Baire. Then the whole union $\bigcup_{s\in S}\mathcal{M}$ is still meager.*

The proof actually requires the axiom of choice. For if $(\mathcal{M}_s)_{s\in S}$ is a partition of $A^\omega$ into mutually disjoint meager sets, then Lemma 3 implies that some union $\bigcup_{s\in T}\mathcal{M}_s$ is not Baire. However, it has been shown by Shelah [She84] that the consistency of *ZFC* implies the consistency of "*ZF* + the axiom of dependent choices + every set in $A^\omega$ is Baire."

*Proof* (cf., [BCGR79]). As $(\mathcal{M}_s)_{s\in S}$ is a mutually disjoint family and $|A^\omega| = |\mathbf{R}|$, we may assume that $S \subseteq \mathbf{R}$. Moreover, we shall assume that $S$ does not contain any uncountable analytic set, e.g., $S$ is a Bernstein set (cf. [Kur66]). Here we need the axiom of choice. Consider the mapping $\Delta : A^\omega \to \mathbf{R}$ which is defined by $\Delta(g) = s$ if $g \in \mathcal{M}_s$ and $\Delta(g) = 0$ otherwise. According to the hypothesis of Lemma 3, the mapping $\Delta$ is a Baire mapping. As $\mathbf{R}$ is separable we may apply Lemma 2. Thus there exists a meager set $\mathcal{M} \subseteq A^\omega$ such that $\Delta \upharpoonright A^\omega \backslash \mathcal{M}$ is a continuous mapping. We may safely assume that $\mathcal{M}$ is a Borel set, hence $A^\omega \backslash \mathcal{M}$ is Borel. So the image $\Delta(A^\omega \backslash \mathcal{M})$ is an analytic subset of $S$ and thus, by choice of $S$, countable.

In other words, $\bigcup_{s\in S}\mathcal{M}_s$ can be written as a countable union of meager sets. $\square$

LEMMA 4. *Let $\Delta : A^\omega \to \mathcal{Y}$ be a Baire mapping, where $\mathcal{Y}$ is a metric space. Then there exists a meager set $\mathcal{M} \subseteq A^\omega$ such that $\Delta \upharpoonright A^\omega \backslash \mathcal{M}$ is continuous.*

*Proof.* The crucial property of metric spaces that we exploit here is that every metric space possesses a $\sigma$-discrete base. This means that there exists a basis $(\mathcal{O}_s^n)_{n<\omega, s\in S_n}$ for $\mathcal{Y}$ such that for each fixed $n < \omega$ the family $(\mathcal{O}_s^n)_{s\in S_n}$ is a family of mutually disjoint sets. Write $\Delta^{-1}(\mathcal{O}_s^n) = \mathcal{B}_s^n \backslash \mathcal{M}_s^n \cup \mathcal{M}_s^n \backslash \mathcal{B}_s^n$, where $\mathcal{B}_s^n \subseteq A^\omega$ is open and $\mathcal{M}_s^n \subseteq A^\omega$ is meager.

As $(\mathcal{O}_s^n)_{s\in S_n}$ is a disjoint family, for every $T \subseteq S_n$ it follows that

$$\bigcup_{s\in T} \Delta^{-1}(\mathcal{O}_s^n) = \left(\bigcup_{s\in T} \mathcal{B}_s^n\right) \Big\backslash \left(\bigcup_{s\in T} \mathcal{M}_s^n\right) \cup \left(\bigcup_{s\in T} \mathcal{M}_s^n\right) \Big\backslash \left(\bigcup_{s\in T} \mathcal{B}_s^n\right),$$

respectively,

$$\left(\bigcup_{s\in T} \mathcal{M}_s^n\right) = \left(\Delta^{-1}\left(\bigcup_{s\in T} \mathcal{O}_s^n\right)\right) \Big\backslash \left(\bigcup_{s\in T} \mathcal{B}_s^n\right) \cup \left(\bigcup_{s\in T} \mathcal{B}_s^n\right) \Big\backslash \left(\Delta^{-1}\left(\bigcup_{s\in T} \mathcal{O}_s^n\right)\right).$$

The right-hand side is a symmetric difference of Baire sets. $\Delta^{-1}(\bigcup_{s\in T}\mathcal{O}_s^n)$ is Baire as $\Delta$ is a Baire mapping and $\bigcup_{s\in T}\mathcal{B}_s^n$ is even open. Hence $\bigcup_{s\in T}\mathcal{M}_s^n$ is Baire.

As the sets $\Delta^{-1}(\mathcal{O}_s^n)$ and $\Delta^{-1}(\mathcal{O}_{s'}^n)$ are disjoint for $s \neq s'$ it follows that $\mathcal{B}_s^n \cap \mathcal{B}_{s'}^n \subseteq (\mathcal{B}_s^n \cap \mathcal{M}_s^n) \cap (\mathcal{B}_{s'}^n \cap \mathcal{M}_{s'}^n)$. But every nonempty open set in $A^\omega$ is nonmeager. Hence, $\mathcal{B}_s^n \cap \mathcal{B}_{s'}^n = \emptyset$. The separability of $A^\omega$ implies that at most countably many $\mathcal{B}_s^n$ are nonempty. In particular, $\bigcup_{s\in T}\mathcal{B}_s^n \cap \mathcal{M}_s^n$ is a meager set for every $T \subseteq S_n$.

The sets $\mathcal{M}_s^n \backslash \mathcal{B}_s^n$ are mutually disjoint. Also, for every $T \subseteq S_n$ the union $\bigcup_{s\in T}\mathcal{M}_s^n \backslash \mathcal{B}_s^n$ is a Baire set, as it merely differs from a Baire set by a meager set, viz.,

$$\left(\bigcup_{s\in T} \mathcal{M}_s^n\right) \Big\backslash \left(\bigcup_{s\in T} \mathcal{M}_s^n \cap \mathcal{B}_s^n\right) \subseteq \bigcup_{s\in T} \mathcal{M}_s^n \Big\backslash \mathcal{B}_s^n \subseteq \bigcup_{s\in T} \mathcal{M}_s^n.$$

Hence $(\mathcal{M}_s^n \setminus \mathcal{B}_s^n)_{s\in S_n}$ satisfies the hypothesis of Lemma 3 and thus

$$\mathcal{M}_n = \left( \bigcup_{s \in S_n} \mathcal{M}_s^n \setminus \mathcal{B}_s^n \right)$$

is a meager set.

Let $\mathcal{M} = \bigcup_{n<\omega} \mathcal{M}_n$. Then, by construction, $\Delta \upharpoonright A^\omega \setminus \mathcal{M}$ is a continuous mapping. □

Putting Lemmas 1 and 4 together yields the following desired reduction.

LEMMA 5. *For every Baire mapping $\Delta : A^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space, there exists an $F \in [A]_{one}\binom{\omega}{\omega}$ such that $\Delta \upharpoonright F\cdot A^\omega$ is continuous.*

**4. Proofs of Theorems A and C.** According to Lemma 5, we may restrict ourselves to continuous mappings $\Delta : A^\omega \to \mathcal{Y}$. The crucial feature of continuity is a kind of *diversification* (cf., [Voi85]).

LEMMA 6. *Let $\Delta_i : A^\omega \to \mathcal{Y}$, $i \in \{0,1\}$, where $\mathcal{Y}$ is a metric space, be two continuous mappings. Then there exists an $F \in [A]_{one}\binom{\omega}{\omega}$ such that either $\Delta_0(F\cdot A^\omega) \cap \Delta_1(F\cdot A^\omega) = \varnothing$, the two mappings have disjoint images on $F\cdot A^\omega$ or such that $\Delta_0(F\cdot g) = \Delta_1(F\cdot g)$ for all $g \in A^\omega$, the two mappings agree on $F\cdot A^\omega$.*

*Proof.* Assume that there exists some $g \in A^\omega$ with $\Delta_0(g) \neq \Delta_1(g)$. As $\Delta_0(g)$ and $\Delta_1(g)$ are distinct elements of a metric space they may be separated by disjoint open sets. Hence, as $\Delta_0$ and $\Delta_1$ are continuous, there exists a positive integer $n$ such that

$$\{\Delta_0(h) \mid h \in A^\omega \text{ and } h(i) = g(i) \text{ for all } i<n\}$$
$$\cap \{\Delta_1(h) \mid h \in A^\omega \text{ and } h(i) = g(i) \text{ for all } i<n\} = \varnothing.$$

Then any $F \in [A]_{one}\binom{\omega}{\omega}$ such that $F(i) = g(i)$ for all $i < n$ has the property that $\Delta_0(F\cdot A^\omega) \cap \Delta_1(F\cdot A^\omega) = \varnothing$. □

*Proof of Theorem* C. According to Lemma 5, we may assume that $\Delta : A^\omega \to \mathcal{Y}$ is a continuous mapping. By induction, using Lemma 6, we construct an $F \in [A]_{one}\binom{\omega}{\omega}$ such that for every $n < \omega$, every $g \in A^n$ and every pair $a, b \in A$ either

$$\{\Delta(F\cdot(g \otimes (a) \otimes h)) \mid h \in A^\omega\} \cap \{\Delta(F\cdot(g\otimes(b)\otimes h)) \mid h \in A^\omega\} = \varnothing \quad \text{or}$$
$$\Delta(F\cdot(g\otimes(a)\otimes h)) = \Delta(F\cdot(g\otimes(b)\otimes h)) \text{ for all } h \in A^\omega.$$

Here, "$\otimes$" denotes the concatenation of sequences. For every such $g \in A^n$, this induces an equivalence relation $\equiv_g$ on $A$. By the Carlson–Simpson Lemma [CS84], there exists an $F' \in [A]_{asc}\binom{\omega}{\omega}$ such that this equivalence relation, say $\equiv$, is the same for all $g \in A^n$, $n < \omega$.

We now claim that for all $g, g' \in A^\omega$ it follows that $\Delta(F\cdot F'\cdot g) = \Delta(F\cdot F'\cdot g')$ if and only if $g \equiv g'$.

First let $g \equiv g'$, i.e., $g(i) \equiv g'(i)$ for all $i < \omega$. By induction on $n$ it follows that $\Delta(F\cdot F'((g(0), \cdots, g(n-1)) \otimes h)) = \Delta(F\cdot F'((g'(0), \cdots, g'(n-1)) \otimes h))$ for all $h \in A^\omega$. Hence, by continuity $\Delta(F\cdot F'\cdot g) = \Delta(F\cdot F'\cdot g')$. Next let $g \not\equiv g'$, say, $n = \min\{i < \omega \mid g(i) \not\equiv g'(i)\}$. As before, we still have that

$$\Delta(F\cdot F'\cdot((g(0), \cdots, g(n-1))\otimes h))$$
$$= \Delta(F\cdot F'\cdot((g'(0), \cdots, g'(n-1))\otimes h)) \text{ for all } h \in A^\omega.$$

As $g(n) \not\equiv g'(n)$ it follows that

$$\{\Delta(F\cdot F'\cdot((g(0), \cdots, g(n-1), g(n))\otimes h)) \mid h \in A^\omega\}$$
$$\cap \{\Delta(F\cdot F'\cdot((g(0), \cdots, g(n-1), g'(n))\otimes h)) \mid h \in A^\omega\} = \varnothing$$

and thus $\Delta(F\cdot F'\cdot g) \neq \Delta(F\cdot F'\cdot g')$; see Fig. 1. □

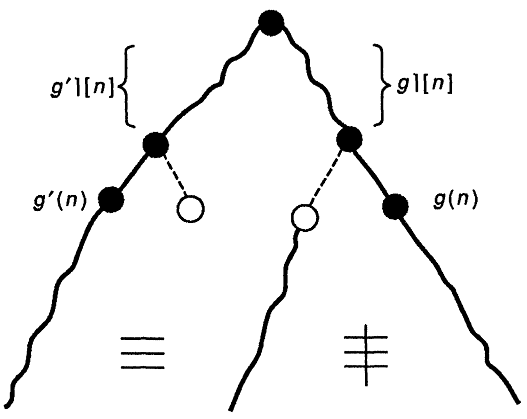


FIG. 1

*Proof of Theorem* A. Again, using Lemma 5, we may assume that $\Delta$ is a continuous mapping. We commence as in the proof of theorem C, using Lemma 6 to construct an $F \in [\{0, 1\}]_{one}\binom{\omega}{\omega}$ such that for every $n < \omega$ and every $g \in \{0, 1\}^n$ either

(i) $$\{\Delta(F\cdot(g \otimes (0) \otimes h)) \mid h \in \{0,1\}^\omega\} \cap \{\Delta(F\cdot(g \otimes (1) \otimes h)) \mid h \in \{0,1\}^\omega\} = \emptyset,$$

or

(ii) $$\Delta(F\cdot(g \otimes (0) \otimes h)) = \Delta(F\cdot(g \otimes (1) \otimes h)) \text{ for all } h \in \{0,1\}^\omega.$$

If there exists some $g \in \bigcup_{n<\omega} \{0, 1\}^n$ such that property (i) holds for all sequences $g \otimes h$, $h \in \bigcup_{n<\omega} \{0, 1\}^n$, then for $G = g \otimes (\lambda_0, \lambda_1, \cdots)$ the restriction $\Delta \upharpoonright F\cdot G\cdot\{0, 1\}^\omega$ is a one-to-one mapping. Otherwise there exists an infinite ascending chain of $g$'s which all have property (ii). In other words, there exists a $g' \in \{0, 1\}^\omega$ and there exists a strictly ascending sequence $(n_i)_{i<\omega}$ such that each $g'\upharpoonright[n_i]$ has property (ii), where $g'\upharpoonright[n_i] = (g'(0), \cdots, g'(n_i))$. Define $G \in [\{0, 1\}]_{one}\binom{\omega}{\omega}$ by

$$G(n_i) = \lambda_i \quad \text{and} \quad G(k) = g'(k) \text{ otherwise.}$$

A simple transitivity argument shows that if for some $h$ and $h \otimes (0) \otimes h'$ property (ii) holds, where $h, h' \in \bigcup_{n<\omega}\{0, 1\}^n$, then property (ii) also holds with respect to $h \otimes (1) \otimes h'$. This argument actually implies that for every $h \in \{0, 1\}^\omega$ and every $i < \omega$ the restriction $(G\cdot h)\upharpoonright[n_i]$ also has property (ii).

Now the same continuity argument as in the proof of Theorem C shows that $\Delta(F\cdot G\cdot h) = \Delta(F\cdot G\cdot h')$ for all $h, h' \in \{0, 1\}^\omega$. $\square$

**5. Proof of Theorem B.** We give an explicit example for $A = \{0, 1, 2\}$. From the construction it will be obvious how to construct analogous mappings for any finite set with more than two elements.

For $g \in \{0, 1, 2\}^\omega$ and $k < \omega$, let pos$(g, k)$ be the position of the $(k + 1)$st 2 in $g$. In other words, $g(\text{pos}(g, k)) = 2$ and $|\{i < \text{pos}(g, k) \mid g(i) = 2\}| = k$.

Now we define $\Delta : \{0, 1, 2\}^\omega \to \{0, 1, 2\}^\omega$ by $\Delta(g) = g'$, where

$$g'(\text{pos}(g,k)) = 2,$$

$$g'(i) = 0 \quad \text{for all } i < \text{pos}(g,0),$$

$$g'(i) = g(i) \quad \text{for all pos}(g,2\cdot k) < i < \text{pos}(g,2\cdot k+1),$$

$$g'(i) = 0 \quad \text{for all pos}(g,2\cdot k+1) < i < \text{pos}(g,2\cdot k+2).$$

For example,

$$\Delta(0,1,1,2,1,0,1,2,1,1,2,0,1,0,2,\cdots)=(0,0,0,2,1,0,1,2,0,0,2,0,1,0,2,\cdots).$$

The mapping $\Delta$ is obviously continuous. It cannot be canonized by any $F \in [\{0, 1, 2\}]_{one}\binom{\omega}{\omega}$, simply because the role of the $(2k)$th, respectively, $(2k+1)$st 2's may not be fixed in advance. Note, however, that for

$$G=(\lambda_0,\lambda_0,\lambda_1,\lambda_1,\lambda_2,\lambda_2,\lambda_3,\lambda_3,\cdots)$$

the restriction $\Delta\upharpoonright G\cdot\{0, 1, 2\}^\omega$ is a constant mapping, whereas for

$$H=(2,\lambda_0,\lambda_0,\lambda_1,\lambda_1,\lambda_2,\lambda_2,\lambda_3,\lambda_3,\cdots)$$

the restriction $\Delta\upharpoonright H\cdot\{0, 1, 2\}^\omega$ is a one-to-one mapping. $\square$

These examples suggest the following question: what are the typical patterns of continuous mappings $\Delta : \{0, 1, 2\}^\omega \to \mathcal{Y}$, where $\mathcal{Y}$ is a metric space, with respect to substructures given by $F \in [\{0, 1, 2\}]_{one}\binom{\omega}{\omega}$?

## REFERENCES

[BCGR79] J. Brzuchowski, J. Cichoń, E. Grzegorek, and C. Ryll-Nardzewski, *On the existence of nonmeasurable unions*, Bull. Acad. Polon. Sci. Ser. Sci. Math., 27 (1979), pp. 447–448.

[Bu79] L. Bukovsky, *Any partition into Lebesgue measurable zero sets produces a non measurable set*, Bull. Acad. Polon. Sci. Ser. Sci. Math., 27 (1979), pp. 431–435.

[CS84] T. J. Carlson and S. G. Simpson, *A dual form of Ramsey's theorem*, Adv. in Math., 53 (1984), pp. 265–290.

[EFK79] A. Emeryk, R. Frankiewicz, and W. Kulpa, *On functions having the Baire property*, Bull. Acad. Polon. Ser. Sci. Math., 27 (1979), pp. 489–491.

[Kur30] K. Kuratowski, *La proprieté de Baire dans les espaces métrique*, Fund. Math., 16 (1930), pp. 390–394.

[Kur66] K. Kuratowski, *Topology*, Vol. I, Academic Press, New York, 1966.

[MS80] G. Moran and D. Strauss, *Countable partitions of product spaces*, Mathematika, 27 (1980), pp. 213–224.

[PS78] K. Prikry and R. M. Solovay, *Images of measures on separable metric spaces*, unpublished manuscript, 1978.

[PSV86] H. J. Prömel, S. G. Simpson, and B. Voigt, *A dual form of Erdös-Rado's canonization theorem*, J. Combin. Theory, Ser. A, 42 (1986), pp. 159–178.

[PV83] H. J. Prömel and B. Voigt, *Canonical partition theorems for parameter sets*, J. Combin. Theory, Ser. A, 35 (1983), pp. 309–327.

[PV85] ———, *Graham Rothschild parameter sets*, WP85403 Institut für Ökonometrie und Operations Research, Universität Bonn, 1985. To appear in *The Mathematics of Ramsey Theory*, J. Nesetril and V. Rödl, eds., Springer-Verlag, New York, Berlin.

[She84] S. Shelah, *Can you take Solovay's inaccessible away*?, Israel J. of Math., 48 (1984), pp. 1–47.

[Voi85] B. Voigt, *Canonizing partition theorems: diversification, products and iterated versions*, J. Combin. Theory, Ser. A, 40 (1985), pp. 349–376.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 530-549, November 1989

# THE AVERAGE NUMBER OF STABLE MATCHINGS*

BORIS PITTEL†

**Abstract.** The probable behavior of an instance of size $n$ of the stable marriage problem, chosen uniformly at random, is studied. The expected number of stable matchings is shown to be asymptotic to $e^{-1}n \ln n$ for $n \to \infty$. The total rank of women by men in the male optimal (pessimal) matching is proved to be close to $n \ln n$ (respectively, $n^2/\ln n$), with high probability.

**Key words.** algorithm, asymptotics, distribution, expectation, stable matching

**AMS(MOS) subject classifications.** primary 05A05, 05C70, 68C25; secondary 60C05, 60F05

**1. Introduction.** In the usual formulation, an instance of size $n$ of the stable marriage problem involves $n$ men and $n$ women, with each person ranking representatives of the opposite sex in order of individual preference for a marriage partner. A complete matching, i.e., a set of $n$ marriages, is called stable if no man and woman, who are not married to each other, would prefer each other to their actual partners under the matching. Gale and Shapley, who introduced the problem, have shown that at least one stable matching does exist; in fact, they provided an iterative procedure that finds a stable set of marriages [4]. Later, McVitie and Wilson [11] developed an alternative ("fundamental") algorithm; its works is described by a sequence of proposals of men to women made one at a time, while the Gale–Shapley algorithm used rounds of simultaneous proposals. Both algorithms yield the same matching that is male-optimal compared to any other stable matching, simultaneously for all men. Using a reduction to a classic urn scheme, Wilson [13] has proved that the expected running time of the fundamental algorithm for the random instance of the problem is bounded by $nH_n$, ($H_n = 1 + \cdots + 1/n$). In the course of a detailed study of the stable marriages problem, Knuth [10] has found a better upper bound $(n-1)H_n + 1$ and has established a lower bound $nH_n - O(\log^4 n)$. Among several open problems, Knuth [10] has posed the question of estimating the expected number of stable matchings. He has indicated that the answer to this question might be found via an integral formula for the probability that a given matching is stable.

A primary purpose of this paper is to establish—by using Knuth's formula—that the expected number of stable matchings is asymptotic to $e^{-1}n \ln n$ for $n \to \infty$. Curiously, it is of the same order as the average number of proposals in the fundamental algorithm. This should be compared with the fact that the *minimum* number of stable matchings for any problem instance of size $n$ is one, while the *maximum* number grows at least exponentially with $n$ (Knuth [10], Irving and Leather [6]). (For other deterministic results on the structure of the set of stable matchings, we refer the reader to Irving [7], Irving, Leather, and Gusfield [8], and Gusfield, Irving, and Leather [5].)

Another purpose of this paper is to show that, almost surely (a.s.) for a random problem instance, the maximum (minimum) total rank of women by men for a stable matching is asymptotic to $n \ln n$ (respectively, $n^2/\ln n$). Since the minimum rank of women by men coincides with the number of proposals by men in the fundamental algorithm, the statement shows that this number is almost surely close to $n \ln n$. On the other hand, the maximum rank of women by men coincides, in distribution, with the total rank of men by women in the male-optimal stable matching that this algorithm determines. So, the latter rank is almost surely close to $n^2/\ln n$, and far exceeds $n \ln n$.

* Received by the editors November 23, 1988; accepted for publication February 1, 1989.

† Department of Mathematics, Ohio State University, Columbus, Ohio 43210.

The stable matching in question heavily favors men at the expense of women. The situation is just the opposite in the female-optimal stable matching.

The rest of the paper is organized as follows. In § 2, we derive a general formula for the probability that a given matching is stable and that its rank has a specified value. This is a generalization of Knuth's formula for the probability that a matching is stable. The latter is used in § 3 to obtain asymptotics of the expected number of stable matchings. The general formula is applied then in § 4 to study the a.s. asymptotic behavior of the minimum rank and the maximum rank for a stable matching. In the Appendix, we prove some auxiliary results for a random partition of the unit interval.

**2. Basic formulas.** By symmetry, each one of $n!$ matchings (pairings) of $n$ men and $n$ women has the same probability $P_n$ of being stable. Knuth [10] has proved that

$$P_n = \overbrace{\int \cdots \int}^{2n} \prod_{1 \leqq i \neq j \leqq n} (1 - x_i y_j)\, dx\, dy \tag{2.1}$$

where $dx = dx_1 \cdots dx_n$, $dy = dy_1 \cdots dy_n$, $0 \leqq x_i \leqq 1$, $0 \leqq y_j \leqq 1$ $(1 \leqq i, j \leqq n)$.

Define the (men-oriented) rank of a stable matching as the sum of the ranks of women by men in this matching. The rank lies between $n$ and $n^2$; it equals $n$ (respectively, $n^2$) if each man happens to be matched with a woman whom he ranks first (respectively, last). Define $P_{nk}$ as the probability that a given matching is stable and that its rank equals $k$ $(n \leqq k \leqq n^2)$. We want to show that

$$P_{nk} = \int \cdots \int [z^{k-n}] \prod_{1 \leqq i \neq j \leqq n} (1 - x_i(1 - z + z y_j))\, dx\, dy; \tag{2.2}$$

here the integrand equals the coefficient of $z^{k-n}$ in the product. Note that this relation implies (2.1) since the sum of the integrands over $k$ equals the integrand in (2.1).

*Proof of* (2.2). (a) Let $U = (u_1, \cdots, u_n)$, $V = (v_1, \cdots, v_n)$ be the sets of men and women, respectively. Each man $u \in U$ (respectively, woman $v \in V$) ranks women (respectively, men) uniformly at random, independently of all other men and women. A way to generate such a random ranking system is as follows. Let us assume that there are given two $n \times n$ matrices $X = [X_{ij}]$, $Y = [Y_{ij}]$ whose entries are all independent, each uniformly distributed on the interval $[0, 1]$. For each man $u_i$ (woman $v_j$) we define a permutation, i.e., ordering, $\pi_i$ (respectively, $\omega_j$) of the set $\{1, \cdots, n\}$ such that

$$X_{i,\pi_i(1)} < X_{i,\pi_i(2)} < \cdots < X_{i,\pi_i(n)}, \qquad (\text{resp., } Y_{\omega_j(1),j} < Y_{\omega_j(2),j} < \cdots < Y_{\omega_j(n),j}).$$

We postulate that the woman $v_{\pi_i(j)}$ is the $j$th best choice for the man $u_i$, and that the man $u_{\omega_j(i)}$ is the $i$th best choice for the woman $v_j$. By the definition of $X$ and $Y$, the $2n$ random permutations are independent of one another, and each is distributed uniformly. (The cases when two elements of one row of $X$, or one column of $Y$, coincide have total probability zero, and thus can be neglected.)

(b) We may, and shall, consider the particular matching

$$M = \{(u_i, v_i) : 1 \leqq i \leqq n\}.$$

The rank $Q_n$ of this matching equals $n + \sum_{i=1}^{n} |\{j : X_{ij} < X_{ii}\}|$, and we need to evaluate $P_{nk}$, the probability of the event $A = \{M \text{ is stable and } Q_n = k\}$.

For $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$ $(0 \leqq x_i, y_j \leqq 1, 1 \leqq i, j \leqq n)$, define $P_{nk}(x, y)$ to be the *conditional* probability of the event $A$ given that $X_{ii} = x_i$, $Y_{jj} = y_j$ $(1 \leqq i \leqq n, 1 \leqq j \leqq n)$; in short $P_{nk}(x, y) = \Pr(A \mid \cdot)$. Since all $X_{\alpha\beta}$, $Y_{\alpha\beta}$ are independent, by the Fubini theorem, it will suffice to show that $P_{nk}(x, y)$ equals the integrand in (2.2). To this end, we first observe that

$$P_{nk}(x,y)=[z^k]E(\chi(M)z^{Q_n} \mid \cdot) \tag{2.3}$$

where $\chi(M)$ is the indicator of the event $\{M \text{ is stable}\}$, and the expected value is conditioned on $X_{ii} = x_i$, $Y_{jj} = y_j$ ($1 \leqq i \leqq n$, $1 \leqq j \leqq n$). To evaluate this expectation, it is convenient to introduce the following "marking" procedure. Fix $z \in (0, 1)$; scan the pairs $(i,j)$ and, whenever $X_{ij} < X_{ii}$ $(=x_i)$, mark the pair with probability $z$, independently of all other pairs. Then, setting

$$B = \{M \text{ is stable } \textit{and} \text{ all the pairs } (i,j) \text{ such that } X_{ij} < X_{ii} \text{ are marked}\},$$

we can write

$$E(\chi(M)z^{Q_n} \mid \cdot) = z^n \Pr(B \mid \cdot). \tag{2.4}$$

Let $C = \{(i,j) : i \neq j\}$, and let $B_{ij}((i,j) \in C)$ be the event

$$\text{"}X_{ii} < X_{ij}, \text{ or } (X_{ii} > X_{ij}, Y_{jj} < Y_{ij} \text{ and } (i,j) \text{ is marked})\text{."}$$

A little reflection shows that

$$B = \bigcap_{(i,j)\in C} B_{ij}.$$

Besides, conditioned on $X_{\alpha\alpha} = x_\alpha$, $Y_{\beta\beta} = y_\beta$ ($1 \leqq \alpha \leqq n$, $1 \leqq \beta \leqq n$), the events $B_{ij}$ are independent, and

$$\Pr(B_{ij} \mid \cdot) = (1-x_i) + x_i(1-y_j)z, \qquad (i,j) \in C.$$

Therefore

$$\Pr(B \mid \cdot) = \prod_{1 \leqq i \neq j \leqq n} (1 - x_i(1 - z + zy_j)),$$

so (see (2.3), (2.4))

$$P_{nk}(x,y) = [z^{k-n}] \prod_{1 \leqq i \neq j \leqq n} (1 - x_i(1 - z + zy_j)). \qquad \square$$

*Note.* To obtain (2.1) directly, rather than from (2.2), we can use a similar argument setting the marking probability $z = 1$, so that $\Pr(B_{ij} \mid \cdot) = 1 - x_i y_j$. The original proof of (2.1) given by Knuth [10] did not use the random matrices $X$, $Y$, but relied instead on an inclusion-exclusion formula, and interpretation of each term as the value of a $2n$-dimensional integral with the integrand equal to the corresponding term in the expansion of $\prod_{1 \leqq i \neq j \leqq n} (1 - x_i y_j)$.

**3. Expected number of stable matchings for large *n*.** We shall prove in this section that

$$P_n = (1 + o(1)) \frac{e^{-1} n \ln n}{n!}. \tag{3.1}$$

Since there are $n!$ matchings, (3.1) immediately implies Theorem 1.

THEOREM 1. *The expected number of stable matchings is asymptotic to* $e^{-1} n \ln n$.

*Proof of* (3.1). In the course of the argument, and in the next section as well, we will use the following facts. Let $X_1, \cdots, X_n$ be independent random variables each distributed uniformly on [0, 1]. Set

$$S_n = \sum_{j=1}^{n} X_j, \qquad T_n = \frac{(\sum_{j=1}^{n} X_j^2)}{S_n^2}.$$

We also introduce the random variables $L_1, \cdots, L_n$ that are the lengths of the consecutive subintervals of [0, 1] obtained by selecting independently $n-1$ points, each uniformly distributed on [0, 1] (in particular, the $L_i$ sum up to one). Set

$$U_n = \sum_{j=1}^{n} L_j^2, \qquad M_n = \max_{1 \leqq j \leqq n} L_j.$$

LEMMA 1. *Let $f_n(\cdot)$, $f_n(\cdot, \cdot)$, $g_n(\cdot)$ be the density of $S_n$, $(S_n, T_n)$, and $U_n$, respectively. Then*

$$f_n(s) = \frac{s^{n-1}}{(n-1)!} \Pr(M_n \leqq s^{-1}), \tag{3.2}$$

*so, in particular,*

$$f_n(s) \leqq \frac{s^{n-1}}{(n-1)!}. \tag{3.3}$$

*Also,*

$$f_n(s,t) \leqq \frac{s^{n-1}}{(n-1)!} g_n(t). \tag{3.4}$$

LEMMA 2 (Asymptotic behavior of $M_n$, $U_n$). *In probability, as $n \to \infty$,*

$$M_n / n^{-1} \ln n \to 1, \qquad n U_n \to 2.$$

*Besides, for every $\rho > 0$,*

$$\Pr(M_n \geqq n^{-1}(\ln n - \ln \ln n - \rho)) \geqq 1 - O(n^{-d}) \quad \forall d \in (0, e^{\rho} - 1).$$

*Note.* The relation (3.2) is well known (see Feller [3, Chap. 1], for instance), but (3.4) appears to be new. We prove both relations in the Appendix by using the fact that the joint density of $L_1, \cdots, L_{n-1}$ equals $(n-1)!$, whenever this density is not zero. As for Lemma 2 (also proved in the Appendix), the argument is based on a classic result as follows: $\{L_j : 1 \leqq j \leqq n\}$ has the same distribution as $\{W_j / \sum_{k=1}^{n} W_k : 1 \leqq j \leqq n\}$, where $W_1, W_2 \cdots$ are independent, exponential, with parameter one (Breiman [1], Karlin and Taylor [9], Rényi [12]). (Sam Karlin has informed me that, in a course he taught in 1986, he used this connection for asymptotic study of $M_n$ and other extremal characteristics of $\{L_j : 1 \leqq j \leqq n\}$.)

(a) Let us begin with an upper bound for $P_n$. Since $1 - \alpha \leqq \exp(-\alpha - \alpha^2/2)$, from (2.1) we get

$$P_n \leqq \overbrace{\int \cdots \int}^{n} \left( \prod_{j=1}^{n} \int_0^1 \exp\left(-y s_j - \frac{y^2 t_j}{2}\right) dy \right) dx_1 \cdots dx_n \tag{3.5}$$

where

$$s_j = \sum_{i \neq j} x_i, \qquad t_j = \sum_{i \neq j} x_i^2,$$

and integration is taken over $0 \leqq x_i \leqq 1$ $(1 \leqq i \leqq n)$.

Fix $a > 0$ and break the integral into two parts, $\int_1$ for $s := x_1 + \cdots + x_n \leqq \omega(n) := a \ln \ln n$ and $\int_2$ for $s > \omega(n)$. Let us show that, for a sufficiently small value of $a$,

$$\int_1 = o\left(\frac{n \ln n}{n!}\right). \tag{3.6}$$

Dropping the factors $\exp(-y^2 t_j/2)$, and integrating with respect to $y$, we have

$$\int_1 \leqq \int \cdots \int_{s \leqq \omega(n)} \left(\prod_{j=1}^{n} \frac{1-\exp(-s_j)}{s_j}\right) dx \qquad (dx = dx_1 \cdots dx_n).$$

To simplify the last estimate, we observe that for $z > 0$,

$$\left(\ln \frac{1-e^{-z}}{z}\right)' = -z^{-1}\left(1 - \frac{z}{e^z - 1}\right),$$

implying existence of a constant $c > 0$ such that

$$-c \leqq \left(\ln \frac{1-e^{-z}}{z}\right)' \leqq 0; \tag{3.7}$$

also

$$\left(\ln \frac{1-e^{-z}}{z}\right)' = -(1+o(1))z^{-1}, \qquad z \to \infty. \tag{3.8}$$

By (3.7),

$$\begin{aligned} \ln \frac{1-e^{-s_j}}{s_j} &= \ln \frac{1-e^{-s}}{s} - \int_{s-x_j}^{s} \left(\ln \frac{1-e^{-z}}{z}\right)' dz \\ &\leqq \ln \frac{1-e^{-s}}{s} + c x_j. \end{aligned}$$

Therefore,

$$\sum_{j=1}^{n} \ln \frac{1-e^{-s_j}}{s_j} \leqq n \ln \frac{1-e^{-s}}{s} + c \sum_{j=1}^{n} x_j,$$

and, since $\sum_j x_j \leqq a \ln \ln n$, we have

$$\int_1 \leqq (\ln n)^{ca} \int_{s \leqq \omega(n)} \left(\frac{1-e^{-s}}{s}\right)^n dx.$$

The last integral is the expected value of $((1-\exp(-S_n))/S_n)^n$ over the event $\{S_n \leqq \omega(n)\}$. Then, by (3.3),

$$\begin{aligned} \int_1 &\leqq (\ln n)^{ca}((n-1)!)^{-1} \int_0^{\omega(n)} \frac{(1-e^{-s})^n}{s} dx \\ &\leqq (\ln n)^{ca}((n-1)!)^{-1} \omega(n) \\ &= O\left(\frac{n(\ln n)^{1/2}}{n!}\right), \end{aligned}$$

provided $a < (2c)^{-1}$. This proves (3.6).

Turning our attention now to $\int_2$, i.e., to $s > \omega(n)$, the generic factor of the product in (3.5) can be estimated as follows:

$$\begin{aligned}\int_0^1 \exp(-ys_j - y^2 t_j/2)\, dy &\leqq s_j^{-1}\int_0^\infty \exp(-z - z^2(t_j s_j^{-2})/2)\, dz\\ &\leqq s_j^{-1}\int_0^\infty \exp(-z-z^2(t_j s^{-2})/2)\, dz\\ &= s_j^{-1}\left(1 - t_j s^{-2}\int_0^\infty z\exp(-z-z^2(t_j s^{-2})/2)\, dz\right)\\ &\leqq s_j^{-1}(1 - t_j s^{-2}F(ts^{-2}))\end{aligned}$$

where

$$(3.9) \qquad t = \sum_{j=1}^n x_j^2, \qquad F(u) = \int_0^\infty z\exp(-z - z^2 u/2)\, dz.$$

Therefore, the integrand in (3.5) is bounded above by

$$\left(\prod_{j=1}^n s_j^{-1}\right)\left(\prod_{k=1}^n (1 - t_k s^{-2}F(ts^{-2}))\right) \leqq \left(\prod_{j=1}^n s_j^{-1}\right)\exp\left(-s^{-2}\left(\sum_{k=1}^n t_k\right)F(ts^{-2})\right).$$

Here

$$\sum_{k=1}^n t_k = (n-1)t,$$

and ($0 \leqq x_j \leqq 1$, $s = \sum_{j=1}^n x_j \geqq \omega(n)$),

$$\begin{aligned}\prod_{j=1}^n s_j^{-1} &= s^{-n}\prod_{j=1}^n\left(1 - \frac{x_j}{s}\right)^{-1}\\ &= s^{-n}\prod_{j=1}^n \exp\left(\frac{x_j}{s} + O\left(\frac{x_j^2}{s^2}\right)\right)\\ &= s^{-n}\exp(1 + O(\omega(n)^{-1})).\end{aligned}$$

Therefore

$$\int_2 \leqq \exp(1 + O(\omega(n)^{-1}))\int_{s \geqq \omega(n)} s^{-n}\exp(-(n-1)ts^{-2}F(ts^{-2}))\, dx;$$

so, using (3.4) of Lemma 1 (and the fact that $s \leqq n$), we have

$$(3.10) \quad \int_2 \leqq \exp(1 + O(\omega(n)^{-1}))((n-1)!)^{-1}\left(\int_{\omega(n)}^n s^{-1}\, ds\right)E(\exp(-(n-1)U_n F(U_n))).$$

Here, by Lemma 2 (see also (3.9)),

$$(3.11) \quad \begin{aligned}(\mathrm{P})\lim_{n\to\infty} \exp(-(n-1)U_n F(U_n)) &= \exp(-2F(0))\\ &= \exp\left(-2\int_0^\infty ze^{-z}\, dz\right) = e^{-2}.\end{aligned}$$

((P) lim designates the limit in probability.) So, invoking the Dominated Convergence Theorem, we can assert that the expected value in (3.10) converges to $e^{-2}$. Besides,

$$\int_{\omega(n)}^{n} s^{-1}\,ds = \ln n(1+O(\ln\ln\ln n/\ln n)). \tag{3.12}$$

Therefore,

$$\int_2 \leqq (1+o(1))\frac{e^{-1}n\ln n}{n!},$$

implying (see (3.6)) that

$$P_n \leqq (1+o(1))\frac{e^{-1}n\ln n}{n!}.$$

(b) It remains to estimate $P_n$ from below. Denote by $D$ the set of all nonnegative $x=(x_1,\cdots,x_n)$ such that

$$3\ln n \leqq s \leqq \frac{n}{\ln^2 n}, \tag{3.13}$$

$$s^{-1}x_j \leqq (1+\varepsilon)\frac{\ln n}{n}, \qquad 1\leqq j\leqq n, \tag{3.14}$$

$$s^{-2}t \leqq (1+\varepsilon)\frac{2}{n}, \tag{3.15}$$

($s=\sum_{j=1}^n x_j$, $t=\sum_{j=1}^n x_j^2$), where $\varepsilon>0$ is fixed. According to (3.13), (3.14), we have

$$x_j \leqq (1+\varepsilon)(\ln n)^{-1} < 1 \qquad (1\leqq j\leqq n) \tag{3.16}$$

for all sufficiently large $n$, so that

$$D\subset\{x:0\leqq x_j\leqq 1, 1\leqq j\leqq n\}.$$

We want to show that the dominant part of $P_n$ is contributed by the region $D$, which is not very surprising in light of Lemmas 1, 2 and part (a). Set

$$P_n(\varepsilon)=\int_{x\in D}\left(\prod_{j=1}^{n}\left(\int_0^1 \prod_{i\neq j}(1-x_iy_j)\,dy_j\right)\right)dx,$$

so that $P_n \geqq P_n(\varepsilon)$. Since $1-\alpha=\exp(-\alpha-\alpha^2(1+O(\alpha))/2)$, $\alpha\to 0$, using (3.16) we can write for fixed $j$,

$$\prod_{i\neq j}(1-x_iy_j)\geqq \exp\left(-y_js_j-y_j^2t_j\frac{1+\sigma_n}{2}\right)$$

where

$$\sigma_n = c\ln^{-1} n, \qquad c=c(\varepsilon)>0.$$

(Recall that $s_j = \sum_{i \neq j} x_i$, $t_j = \sum_{i \neq j} x_i^2$.) Hence, for each $j$,

$$\begin{aligned} I_j(x) &:= \int_0^1 \prod_{i \neq j} (1 - x_i y_j)\, dy_j \\ &\geqq \int_0^1 \exp\left(-ys_j - y^2 t_j \frac{1+\sigma_n}{2}\right) dy \\ &= s_j^{-1} \int_0^{s_j} e^{-z} \exp\left(-z^2 (t_j s_j^{-2} \frac{1+\sigma_n}{2}\right) dz \\ &\geqq \frac{1-e^{-s_j}}{s_j} \exp\left(-(t_j s_j^{-2}) \frac{1+\sigma_n}{2(1-e^{-s_j})} \int_0^{s_j} e^{-z} z^2\, dz\right). \end{aligned} \tag{3.17}$$

(In the last step, we have used Jensen's inequality, namely that

$$\int_{z_1}^{z_2} A(z) B(C(z))\, dz \geqq B\left(\int_{z_1}^{z_2} A(z) C(z)\, dz\right),$$

if $B(\cdot)$ is convex, $A(z) \geqq 0$, and $\int_{z_1}^{z_2} A(z)\, dz = 1$. To simplify this estimate, we observe that, by the definition of $D$,

$$\begin{aligned} s_j &= s - x_j = s\left(1 - \frac{x_j}{s}\right) \\ &= s \exp\left(-\frac{x_j}{s} + O\left(\frac{x_j^2}{s^2}\right)\right) = s \exp\left(-\frac{x_j}{s} + o\left(\frac{x_j}{s}\right)\right) \\ &\geqq c' \ln n \quad \forall c' \in (2,3), \end{aligned}$$

uniformly over $x \in D$, if $n$ is large. Therefore

$$\begin{aligned} 1 - e^{-s_j} &= 1 + O(n^{-c'}), \\ \int_0^{s_j} e^{-z} z^2\, dz &= 2 - \frac{e^{-s_j}}{2}\left(1 + s_j + \frac{s_j^2}{2}\right) \\ &= 2 + O(n^{-c''}), \qquad c'' = c' - 2 > 0, \end{aligned}$$

and (3.17) becomes

$$I_j(x) \geqq (1 + O(n^{-c'})) s^{-1} \exp\left(\frac{x_j}{s} + o\left(\frac{x_j}{s}\right)\right) \exp(-(t_j s_j^{-2})(1 + O(\ln^{-1} n))).$$

But $c' > 2$, $s_j \leqq s = \sum_{j=1}^n x_j$, $\sum_{j=1}^n t_j = (n-1)t$; therefore

$$\prod_{j=1}^n I_j(x) \geqq (1 + o(1)) e s^{-n} \exp(-n(ts^{-2})(1 + O(\ln^{-1} n))) \tag{3.18}$$

uniformly over $x \in D$.

Let us switch to new variables, namely

$$u = \sum_{j=1}^n x_j = s, \qquad v_j = x_j s^{-1}, \quad 1 \leqq j \leqq n-1.$$

Define also $v_n = x_n s^{-1}$. Clearly, $0 \leqq v_j \leqq 1$ and $\sum_{j=1}^n v_j = 1$. The conditions (3.13)–

(3.15), that define $D$ become

$$3 \ln n \leqq u \leqq \frac{n}{\ln^2 n}, \tag{3.19}$$

$$v_j \leqq (1+\varepsilon)\frac{\ln n}{n}, \tag{3.20}$$

$$\sum_{j=1}^{n} v_j^2 \leqq (1+\varepsilon)\frac{2}{n}. \tag{3.21}$$

Thus, in new variables, the region is the *direct product* of the interval defined in (3.19) and a region $D^*$ defined in (3.20), (3.21). Besides, the Jacobian of $(x_1, \cdots, x_n)$ with respect to $(u, v_1, \cdots, v_{n-1})$ equals $u^{n-1}$. So, it follows from the estimate (3.18) that

$$P_n(\varepsilon) \geqq (1+o(1))e((n-1)!)^{-1}\left(\int_{3\ln n}^{n/\ln^2 n} u^{-1}\,du\right) \tag{3.22}$$

$$\cdot \int_{D^*} \exp\left(-n(1+O(\ln^{-1} n))\sum_{j=1}^{n} v_j^2\right)(n-1)!\,dv, \qquad dv = dv_1 \cdots dv_{n-1}.$$

The first integral here is $\ln n(1+o(1))$. Let us have a closer look at the second integral. Its integrand is at least $\exp(-2(1+\varepsilon)(1+O(\ln^{-1} n)))$ everywhere on $D^*$. In addition, $(n-1)!$ is the joint density of the first $(n-1)$ subintervals among $n$ subintervals $L_1, \cdots, L_n$ introduced in Lemma 1 (see Breiman [1, Chap. 13], for instance). Thus, the inequality (3.21) leads to

$$P_n(\varepsilon) \geqq (1+o(1))\exp(-1-2\varepsilon)\frac{n \ln n}{n!}$$

$$\cdot \Pr\left(\max_{1 \leqq j \leqq n} L_j \leqq (1+\varepsilon)\frac{\ln n}{n}, \sum_{j=1}^{n} L_j^2 \leqq (1+\varepsilon)\frac{1}{n}\right)$$

for every fixed $\varepsilon > 0$. But the probability of the event on the right-hand side tends to one as $n \to \infty$ (see Lemma 2). Letting $n \to \infty$ and then $\varepsilon \downarrow 0$, we are able to conclude that

$$P_n \geqq (1+o(1))\frac{e^{-1} n \ln n}{n!}.$$

## 4. Probable behavior of the maximum rank and the minimum rank of a stable matching.

Let $r_n$ and $R_n$ be the minimum (*male related*) rank and the maximum rank for any stable matching. It is well known that $r_n$ equals the total number of proposals in the fundamental algorithm in which men propose to women. The resulting stable matching is both male optimal and female pessimal. So, by symmetry, we can assert that, *in distribution*, $R_n$ coincides with the *female related* rank of that particular matching.

Our goal is to prove the following theorem.

THEOREM 2. *In probability*

$$\frac{r_n}{n \ln n} \to 1, \tag{4.1}$$

$$\frac{R_n}{n^2 \ln^{-1} n} \to 1, \tag{4.2}$$

*as* $n \to \infty$.

*Notes*. According to this theorem, the stable matching reached via proposals made by men to women is almost surely considerably more favorable to men than to women. In short, initiative pays! Also, the relation (4.1) means that the number of proposals in the fundamental algorithm is almost surely close to $n \ln n$.

The core of the proof is the following statement.

PROPOSITION. *For every* $\rho > 0$ *and* $\delta \in (0, e^\rho - 1)$,

$$\Pr(r_n \geqq n(\ln n - \ln\ln n - \rho)) \geqq 1 - O(n^{-\delta}), \tag{4.3}$$

$$\Pr(R_n \leqq n^2 \ln^{-1} n(1 + (\ln\ln n + \rho)\ln^{-1} n)) \geqq 1 - O(n^{-\delta}). \tag{4.4}$$

*Proof.* Note first that for every $k$ between $n$ and $n^2$,

$$\Pr(r_n \leqq k) \leqq n! \sum_{m=n}^{k} P_{nm}, \qquad \Pr(R_n \geqq k) \leqq n! \sum_{m=k}^{n^2} P_{nm}.$$

Here $P_{nm}$ is the probability that a fixed matching is stable and its rank equals $m$. In § 2, we have proved that

$$\begin{aligned} P_{nm} &= \iint ([z^{m-n}]\Phi(x,y,z))\,dx\,dy, \\ \Phi(x,y,z) &:= \prod_{i \neq j} (1 - x_i(1 - z + zy_j)). \end{aligned} \tag{4.5}$$

Mimicking an approach due to Chernoff [2] (that allows us to estimate the tails of a distribution through its moment generating function), we can then write

$$\Pr(r_n \leqq k) \leqq n! \iint \inf_{0<z\leqq 1} (z^{n-k}\Phi(x,y,z))\,dx\,dy, \tag{4.6}$$

$$\Pr(R_n \geqq k) \leqq n! \iint \inf_{z \geqq 1} (z^{n-k}\Phi(x,y,z))\,dx\,dy. \tag{4.7}$$

In the following argument we will not try to determine the best $z = z(x, y)$; it will be sufficient to choose $z = z(s)$ $(s = \sum_{i=1}^n x_i)$.

(I) Consider $r_n$ first. Bounding each factor $1 - x_i(1 - z + zy_j)$ in (4.5) by $\exp(-x_i(1 - z + zy_j))$, and integrating with respect to $y = (y_1, \cdots, y_n)$, from (4.6) we obtain

$$\Pr(r_n \leqq k) \leqq n! \int \inf_{0<z\leqq 1} \left( z^{n-k} \exp((z-1)(n-1)s) \prod_{j=1}^{n} \frac{1 - e^{-zs_j}}{zs_j} \right) dx,$$

$(s_j = \sum_{i \neq j} x_i)$. Here, by (3.7), (3.8), and the condition $z \leqq 1$, we have

$$\prod_{j=1}^{n} \frac{1 - e^{-zs_j}}{zs_j} \leqq c \left( \frac{1 - e^{-zs}}{zs} \right)^n$$

where $c$ is an absolute constant. In conjunction with (3.3) of Lemma 1, we then have

$$\Pr(r_n \leqq k) \leqq cn \int_0^n \inf_{0<z\leqq 1} (\exp(H(s,z)))\,ds, \tag{4.8}$$

$$H(s,z) := (z-1)(n-1)s + n\ln(1 - e^{-zs}) - k\ln z - \ln s \tag{4.9}$$

for all $n \leqq k \leqq n^2$. The relation (4.3) will be proved when we show that the right-hand expression in (4.8) goes to zero as $n^{-\delta}$ ($\delta \in (0, e^{\rho} - 1)$), for

$$k = n(\ln n - \ln\ln n - \rho). \tag{4.10}$$

To make the best use of (4.8), it is natural to choose $z = z(s)$ that minimizes $H(s, z)$ for $z \in (0, 1]$. But

$$H_z = (n-1)s + ns(e^{zs} - 1)^{-1} - kz^{-1} = 0$$

if $zs = a$ and $a$ satisfies an equation

$$h(\alpha) = k, \qquad h(\alpha) := \alpha((n-1) + n(e^{\alpha} - 1)^{-1}). \tag{4.11}$$

Now, $h(0+) = n$, and an elementary (albeit tedious) computation shows that

$$h'(\alpha) \geqq h'(0+) = \frac{n}{2} - 1.$$

Hence, (4.11) does have a unique positive root $a = a(k)$ for all $k > n$, and $a(k)$ is continuously differentiable with $a'(k) > 0$. In particular, if $k$ is given by (4.10), then

$$\begin{aligned} a &= \left(1 + O\left(\frac{\ln n}{n}\right)\right)\left(\frac{k}{n}\right) \\ &= \ln n - \ln\ln n - \rho + O\left(\frac{\ln^2 n}{n}\right) \\ &< \ln n - \ln\ln n - \rho' \quad \forall \rho' < \rho \end{aligned} \tag{4.12}$$

for $n$ sufficiently large.

Now, we can choose $z = a/s$ if $s > a$, and $z \equiv 1$ for $s \leqq a$. Then from (4.9) it follows that

$$\begin{aligned} &\int_0^n \inf_{0<z\leqq 1} (\exp(H(s,z)))\, ds \\ &\qquad \leqq \int_0^a s^{-1}(1 - e^{-s})^n\, ds + a^{-k}(1 - e^{-a})^n e^{(n-1)a} \int_a^{\infty} s^{k-1} e^{-(n-1)s}\, ds \\ &\qquad \leqq A_1 + A_2 \end{aligned}$$

where

$$A_1 = \int_0^a s^{-1}(1 - e^{-s})^n\, ds, \qquad A_2 = (a(n-1))^{-k}(1 - e^{-a})^n e^{(n-1)a}(k-1)!.$$

So, if $k$ satisfies (4.10), then

$$\begin{aligned} A_1 &\leqq a(1 - e^{-a})^{n-1} = O(a \exp(-ne^{-a})) \\ &= O(n^{-e^{\rho'}}) \quad \forall \rho' \in (0, \rho). \end{aligned} \tag{4.13}$$

Furthermore, by Stirling's formula for factorials, we have

$$A_2 = O\left(\left(\frac{\ln n}{n}\right)^{1/2} \exp(\phi_n(k))\right), \tag{4.14}$$

where

$$(4.15)\qquad \phi_n(k) = F_n(k, a(k)),$$

$$(4.16)\qquad F_n(\kappa, \alpha) := (n-1)\alpha + n\ln(1-e^{-\alpha}) - \kappa\ln\alpha + (\kappa-1)\ln\frac{\kappa-1}{e(n-1)}.$$

To sharply bound $\phi_n(k)$ from above, we need to look closer at $F_n(\kappa, \alpha)$. First,

$$\frac{\partial F_n(\kappa,\alpha)}{\partial\alpha} = (n-1) + n(e^{\alpha}-1)^{-1} - \kappa\alpha^{-1},$$

so that

$$(4.17)\qquad \left.\frac{\partial F_n(\kappa,\alpha)}{\partial\alpha}\right|_{\alpha=a(\kappa)} = 0$$

(see (4.11)) and

$$(4.18)\qquad \frac{\partial F_n(\kappa,\alpha)}{\partial\kappa} = \ln\frac{\kappa-1}{\alpha(n-1)}.$$

Consequently, $(\kappa_0, \alpha_0)$ defined by

$$(4.19)\qquad e^{\alpha_0} - 1 = n\alpha_0,$$

$$(4.20)\qquad \kappa_0 = (n-1)\alpha_0 + 1$$

is a stationary point of $F_n(\kappa, \alpha)$. An easy bootstrapping shows that

$$(4.21)\qquad \alpha_0 = \ln n + \ln\ln n(1 + O(\ln^{-1} n)), \qquad \kappa_0 = n(\ln n + \ln\ln n(1 + O(\ln^{-1} n))),$$

so that $\kappa_0 > k$ (see (4.10)). Then, by (4.15), (4.16), and (4.19)-(4.21), we have

$$(4.22)\qquad \begin{aligned} \phi_n(\kappa_0) &= F_n(\kappa_0, \alpha_0) = n\ln(1-e^{-\alpha_0}) - \ln\alpha_0 \\ &= -\ln\alpha_0 + O(ne^{-\alpha_0}) = -\ln\alpha_0 + O(\ln^{-1} n). \end{aligned}$$

Let us show that, in fact, $\phi_n(\kappa_0) = \max\phi_n(\kappa)$. Indeed, since $a = a(\kappa)$ is differentiable, using (4.17), (4.18), we obtain

$$(4.23)\qquad \begin{aligned} \phi'_n(\kappa) &= \left.\frac{\partial F_n(\kappa,\alpha)}{\partial\kappa}\right|_{\alpha=a=a(\kappa)} \\ &= \ln\left(1 + \frac{1}{a(n-1)}\left(\frac{na}{e^a-1} - 1\right)\right) > (<) 0, \end{aligned}$$

for $\kappa < (>) \kappa_0$. ($\alpha/(e^{\alpha}-1)$ decreases if $\alpha$ increases, and $a'(\kappa) > 0$.)

Since $\kappa < \kappa_0$, we now know that $\phi_n(k) < \phi_n(\kappa_0)$. Still, we would like to do better, knowing also that, in fact, $\kappa_0 - k$ is close to $2n\ln\ln n$. To this end, choose $\kappa_1$ such that $\alpha_1 = a(\kappa_1)$ satisfies

$$(4.24)\qquad \frac{n\alpha_1}{e^{\alpha_1}-1} = \ln\ln n,$$

that is (cf. (4.19), (4.21)),

$$(4.25)\qquad \alpha_1 = \ln n + \ln\ln n(1 + O(\ln^{-1} n)).$$

Then $\kappa_1 \in (k, \kappa_0)$; really, $\kappa_1 < \kappa_0$ since $\alpha_1 < \alpha_0$ (compare (4.21) and (4.25)) and $k < \kappa_1$ (compare (4.12) and (4.25)). Therefore,

$$\begin{aligned}\phi_n(k) &= \phi_n(\kappa_0) - \int_k^{\kappa_0} \phi_n'(\kappa)\, d\kappa \\ &\leqq \phi_n(\kappa_0) - \int_k^{\kappa_1} \phi_n'(\kappa)\, d\kappa.\end{aligned} \tag{4.26}$$

The derivative $\phi_n'(\kappa)$ is given by (4.23). Since $na/(e^a - 1)$ is at least $\ln \ln n$ (see (4.24)) on $[k, \kappa_1]$, we easily get

$$\phi_n'(\kappa) = (1 + o(1))e^{-a}, \qquad a = a(\kappa), \tag{4.27}$$

uniformly over $\kappa \in [k, \kappa_1]$. Besides, considering $\kappa$ as a function of $a$, we have (see (4.11)) also

$$\begin{aligned}\frac{d\kappa}{da} &= (n-1) + n\frac{e^a - 1 - ae^a}{(e^a - 1)^2} \\ &= n(1 + O(ae^{-a})) = n(1 + o(1)).\end{aligned} \tag{4.28}$$

Combining (4.26)–(4.28), we estimate

$$\phi_n(k) \leqq \phi_n(\kappa_0) - (1 + o(1))n(e^{-a(k)} - e^{-\alpha_1}).$$

Here (as in (4.13)),

$$ne^{-a(k)} \geqq e^{\rho'} \ln n \quad \forall \rho' \in (0, \rho),$$

and, by (4.22), (4.25),

$$\begin{aligned}&\phi_n(\kappa_0) \leqq 0, \\ ne^{-\alpha_1} &= n \exp(-\ln n - \ln \ln n(1 + O(\ln^{-1} n))) \\ &= O(\ln^{-1} n).\end{aligned}$$

So, we arrive at

$$\phi_n(k) \leqq -e^{\rho'} \ln n \quad \forall \rho' \in (0, \rho). \tag{4.29}$$

Therefore (see (4.14))

$$A_2 = O(n^{-(1/2 + e^{\rho'})}) \quad \forall \rho' \in (0, \rho). \tag{4.30}$$

The estimates (4.8), (4.13), and (4.30) show that

$$\Pr(r_n \leqq n(\ln n - \ln \ln n - \rho)) = O(n^{-(e^{\rho'} - 1)}) \quad \forall \rho' \in (0, \rho). \qquad \Box$$

(II) Turn now to $R_n$, the maximum rank of a stable matching. With the help of (3.2) from Lemma 1, (3.7), (3.8), and (4.7), we obtain similarly to (4.8), (4.9),

$$\Pr(R_n \geqq k) \leqq cn \int_0^n \inf_{z \geqq 1} (\exp(H_1(s, z)))\theta_n(s)\, ds \tag{4.31}$$

where

$$\theta_n(s) = \Pr(M_n \leqq s^{-1}),$$

and

$$(4.32)\qquad H_1(s,z)=\begin{cases} H(s,z)+\gamma sz, & s\leqq s_0,\\ H(s,z), & s>s_0;\end{cases}$$

$\gamma$, $s_0$ are absolute constants. (In part (I) we could afford to drop the factor $\theta_n(s)$, but this time we will need it.)

Let us show that the right-hand expression in (4.31) goes to zero as $n^{-\delta}$ ($\delta \in (0, e^\rho - 1)$), for

$$k=n^2 \ln^{-1} n(1+(\ln\ln n+\rho)\ln^{-1} n).$$

The root $a$ of (4.11) is this time

$$a=n\ln^{-1} n(1+(\ln\ln n+\rho_n)\ln^{-1} n),$$

where

$$\rho_n=\rho+o(1).$$

We choose $z = a/s$ if $s < a$, and $z \equiv 1$ for $s \geqq a$. (Recall that a feasible $z$ must satisfy $z \geqq 1$.) Set

$$(4.33)\qquad a_1=n\ln^{-1} n(1+(\ln\ln n+\rho')\ln^{-1} n),\qquad \rho'\in(0,\rho).$$

Breaking $[0, n]$ into $[0, a_1]$, $[a_1, a]$, and $[a, n]$, and using (4.9), (4.32), we can write

$$\int_0^n \inf_{z\geqq 1}(\exp(H_1(s,z)))\theta_n(s)\,ds\leqq a^{-k}(1-e^{-a})^n e^{(n-1)a}(B_1+B_2)+B_3$$

where

$$B_1=e^{\gamma a}\int_0^{a_1} s^{k-1}e^{-(n-1)s}\theta_n(s)\,ds,$$

$$B_2=\int_{a_1}^{a} s^{k-1}e^{-(n-1)s}\theta_n(s)\,ds,$$

$$B_3=\int_a^n s^{-1}(1-e^{-s})^n\theta_n(s)\,ds.$$

We estimate $B_1$, $B_2$, $B_3$, moving backward.

(1) For $s\in[a, n]$,

$$\theta_n(s)=\Pr(M_n\leqq s^{-1})\leqq \Pr(M_n\leqq a^{-1})=\theta_n(a).$$

Since

$$a^{-1}=n^{-1}(\ln n-\ln\ln n-\rho_n'),\qquad \rho_n'=\rho+o(1),$$

then by Lemma 2 we have

$$(4.34)\qquad B_3\leqq\theta_n(a)\int_a^n s^{-1}\,ds=O(n^{-e^{\rho_3}})\quad \forall\rho_3\in(0,\rho).$$

(2) Next,

$$B_2\leqq\theta_n(a_1)\int_{a_1}^{a} s^{k-1}e^{-(n-1)s}\,ds.$$

Here

$$\theta_n(a_1)=O(n^{-e^{\rho_2}}) \quad \forall \rho_2\in(0,\rho'),$$

because (see (4.33)),

$$a_1^{-1}=n^{-1}(\ln n-\ln\ln n-\rho'+o(1)).$$

Also

$$\int_{a_1}^{a} s^{k-1}e^{-(n-1)s}\,ds\leqq\int_0^\infty s^{k-1}e^{-(n-1)s}\,ds$$
$$=(n-1)^{-k}(k-1)!=O\left(\left(\frac{k-1}{e(n-1)}\right)^{k-1}\right).$$

($k\sim n^2\ln^{-1}n$.) Therefore,

$$B_2=O\left(n^{-e^{\rho_2}}\left(\frac{k-1}{e(n-1)}\right)^{k-1}\right). \tag{4.35}$$

(3) Finally,

$$B_1\leqq e^{\gamma a}\int_0^{a_1} s^{k-1}e^{-(n-1)s}\,ds$$
$$=e^{\gamma a}\int_0^{a_1} e^{q(s)}\,ds, \qquad q(s):=(k-1)\ln s-(n-1)s.$$

The function $q(s)$ achieves its maximum at

$$s_*=\frac{k-1}{n-1}=n\ln^{-1}n(1+(\ln\ln n+\rho_n'')\ln^{-1}n), \qquad \rho_n''=\rho+o(1).$$

By (4.33), $s_*>a_1$, so that $q(s)$ is increasing on $[0,a_1]$. Also

$$s_*=a_1(1+o(1)), \qquad s_*-a_1=n\ln^{-2}n(\rho-\rho'+o(1)).$$

Therefore, since $q''(s)=-s^{-2}(k-1)$,

$$B_1\leqq e^{\gamma a}a_1e^{q(a_1)}\leqq a_1e^{\gamma a}\exp\left(q(s_*)+\frac{1}{2}q''(\tilde{s})(a_1-s^*)^2\right) \qquad (\tilde{s}\in[a_1,s^*])$$

$$=O\left(\left(\frac{k-1}{e(n-1)}\right)^{k-1}\exp(\gamma a-\lambda n^2\ln^{-3}n)\right) \tag{4.36}$$
$$=O\left(\left(\frac{k-1}{e(n-1)}\right)^{k-1}\exp(-\lambda n^2\ln^{-3}n/2)\right), \qquad \lambda>0.$$

(Recall that $a=O(n\ln^{-1}n)$, so that $a=o(n^2\ln^{-3}n)$.)

A combination of (4.31), (4.34)–(4.36) at last yields

$$\Pr(R_n\leqq k)\leqq cn(n^{-e^{\rho_2}}\exp(\phi_n(k))+n^{-e^{\rho_3}}),$$

for every $\rho_2\in(0,\rho')$ and every $\rho_3\in(0,\rho)$, provided that $\rho'<\rho$. Here, as in part (I),

$$\phi_n(k)=F_n(k,a), \qquad a=a(k).$$

It remains to recall (see (4.22)) that the maximum value of the function $\phi_n(k)$ is negative.

Proof of the proposition is now complete. □

The rest is short. First, $r_n$ equals the total number of proposals in the fundamental algorithm, and this number is *stochastically* dominated (Knuth [10], Wilson [13]) by the total number of draws in the coupon collector problem with $n$ coupons, and the expected value of the latter is $nH_n$ ($H_n = 1 + 1/2 + \cdots + 1/n$).

Given $\varepsilon_n > 0$, let $A$ be the event that a certain coupon, say $j$, has not been drawn in the first $N = (1 + \varepsilon_n)n \ln n$ draws. Then

$$\begin{aligned}\Pr(r_n \leqq (1+\varepsilon_n)n \ln n) &\geqq 1 - n \Pr(A)\\ &= 1 - n(1-n^{-1})^N \geqq 1 - n \exp(-N/n)\\ &= 1 - n^{-\varepsilon_n} \to 1,\end{aligned}$$

provided that $\varepsilon_n \ln n \to \infty$. Combining this with (4.3), we obtain

$$\Pr(n(\ln n - \ln\ln n - \rho) \leqq r_n \leqq n(\ln n + \omega(n))) \to 1,$$

for every $\rho > 0$, and $\omega(n) \to \infty$ however slowly. Consequently, $r_n/n \ln n \to 1$ in probability.

Furthermore, denote by $\pi_{nj}$ the total number of proposals made to a woman $j$ in the course of the fundamental algorithm. Let $R_{nj}$ be the rank of her eventual partner, according to her preferences, needless to say. By symmetry,

$$E(\pi_{n1}) = \cdots = E(\pi_{nn}) \leqq n^{-1}(nH_n) = H_n. \tag{4.37}$$

Besides, given $\pi_{nj} = k$, $R_{nj} - 1$ is binomially distributed with parameters $n - k$ and $p = (k+1)^{-1}$. Therefore

$$E(R_{nj} \mid \pi_{nj}) = 1 + \frac{n - \pi_{nj}}{\pi_{nj}+1} = \frac{n+1}{\pi_{nj}+1},$$

and, by Jensen's inequality and (4.37),

$$E(R_{nj}) \geqq \frac{n+1}{E(\pi_{nj})+1} = \frac{n+1}{H_n+1}.$$

So,

$$\begin{aligned}E(R_n) = E\left(\sum_{j=1}^{n} R_{nj}\right) &\geqq n\frac{n+1}{H_n+1}\\ &= n^2 \ln^{-1} n(1 + O(\ln^{-1} n)).\end{aligned}$$

A simple argument that uses (4.4) and the last relation yields that, in probability, $R_n/n^2 \ln^{-1} n \to 1$.

Theorem 2 is proved.

## Appendix.

*Proof of Lemma* 1. For $0 < s_1 < s_2 < n$,

$$\Pr(s_1 \leqq S_n \leqq s_2) = \int \cdots \int dx_1 \cdots dx_n$$

where $0 \leqq x_j \leqq 1$ $(1 \leqq j \leqq n)$, and $s_n = \sum_{j=1}^n x_j \in (s_1, s_2)$. We switch to new variables:

$$u = \sum_{j=1}^{n} x_j, \qquad v_j = x_j u^{-1}, \quad 1 \leqq j \leqq n-1.$$

Define also $v_n = x_n u^{-1}$, so that $\sum_{j=1}^n v_j = 1$. The inverse transformation is

$$x_j = u v_j, \qquad 1 \leqq j \leqq n$$

where

$$v_n = 1 - \sum_{j=1}^{n-1} v_j.$$

Its Jacobian is $u^{n-1}$, whence

$$\Pr(s_1 \leqq S_n \leqq s_2) = \int \cdots \int u^{n-1}\, du\, dv_1 \cdots dv_{n-1}$$

where $s_1 \leqq u \leqq s_2$ and $\max_{1 \leqq j \leqq n} v_j \leqq u^{-1}$. Therefore,

$$f_n(s) = s^{n-1} \underbrace{\int \cdots \int}_{\substack{\max v_j \leqq s^{-1} \\ 1 \leqq j \leqq n}} dv_1 \cdots dv_{n-1}$$

$$= \frac{s^{n-1}}{(n-1)!} \Pr(M_n \leqq s^{-1}),$$

since $(n-1)!$ is the joint density of the first $(n-1)$ intervals $L_1, \cdots, L_{n-1}$ in the random partition of $[0, 1]$ by $(n-1)$ random points.

Similarly, denoting $\sum_{j=1}^n v_j^2$ by $t$,

$$\Pr(s_1 \leqq S_n \leqq s_2, t_1 \leqq T_n \leqq t_2) = \underbrace{\int \cdots \int}_{\substack{s_1 \leqq u \leqq s_2 \\ t_1 \leqq t \leqq t_2 \\ \max v_j \leqq u^{-1}}} u^{n-1}\, du\, dv_1 \cdots dv_{n-1}$$

$$\leqq \underbrace{\int \cdots \int}_{\substack{s_1 \leqq u \leqq s_2 \\ t_1 \leqq t \leqq t_2}} u^{n-1}\, du\, dv_1 \cdots dv_{n-1}$$

(in both integrals, $\sum_{j=1}^{n-1} v_j \leqq 1$). Since this inequality holds for all $s_1 < s_2$ and $t_1 < t_2$, we have

$$f_n(s, \tau) \leqq \frac{s^{n-1}}{(n-1)!} \frac{d}{d\tau}\Bigg(\underbrace{\int \cdots \int}_{t \leqq \tau} (n-1)!\, dv_1 \cdots dv_{n-1}\Bigg)$$

$$= \frac{s^{n-1}}{(n-1)!} g_n(\tau)$$

where $g_n(\cdot)$ is the density of $\sum_{j=1}^n L_j^2$. □

*Proof of Lemma* 2. The random variables $L_1, \cdots, L_n$ are exchangeable and, for $1 \leqq k \leqq n-1$, the joint density of $L_1, \cdots, L_k$ is given by

$$f_{L_1 \cdots L_k}(x_1, \cdots, x_k) = (n-1)^{\underline{k}}\left(1-\sum_{j=1}^{k} x_j\right)^{n-k-1}, \tag{A1}$$

$$\left(a^{\underline{b}} \underset{\text{def}}{=} a(a-1)\cdots(a-b+1)\right)$$

where $0 \leqq x_j \leqq 1$, $\sum_{j=1}^{k} x_j \leqq 1$. In particular,

$$\Pr(L_1 \geqq x_1, \cdots, L_k \geqq x_k) = \left(1-\sum_{j=1}^{k} x_j\right)^{n}, \tag{A2}$$

provided that $\sum_{j=1}^{k} x_j \leqq 1$. (See Feller [3, Chap. 1].)

(1) Fix $z$ and let $x = (\ln n + z)/n$. Define $N_n$ as the total number of the variables $L_j \geqq x$. Then by (A2), for every $k \geqq 1$ and $n$ large enough,

$$\begin{aligned} E(N_n^{\underline{k}}) &= n^{\underline{k}} \Pr(L_1 \geqq x, \cdots, L_k \geqq x) \\ &= n^{\underline{k}}(1-kx)^n = (1+o(1))n^k \exp(n \ln(1-kx)) \\ &= (1+o(1)) \exp\left(k \ln n + n\left(-kx + O\left(\frac{\ln^2 n}{n^2}\right)\right)\right) \\ &= (1+o(1))(e^{-z})^k, \qquad n \to \infty. \end{aligned}$$

Therefore, $N_n$ converges in distribution to a Poisson distributed random variable $N$ with parameter $\lambda = e^{-z}$. Consequently, denoting $\max_{1 \leqq j \leqq n} L_j$ by $M_n$,

$$\Pr\left(M_n < \frac{\ln n + z}{n}\right) = \Pr(N_n = 0) \to \Pr(N=0) = e^{-e^{-z}},$$

and

$$M_n = \frac{\ln n + O_p(1)}{n},$$

where $O_p(1)$ stands for a random variable bounded in probability as $n \to \infty$. □

(2) Let $x = (\ln n - \ln \ln n - \rho)/n$. We want to show that

$$\Pr(M_n \leqq x) = O(n^{-e^{\rho'}}) \quad \forall \rho' \in (0, \rho).$$

To this end, we observe that $(L_1, \cdots, L_n)$ coincides in distribution with $(\mathscr{L}_1, \cdots, \mathscr{L}_n)$ where

$$\mathscr{L}_j = W_j\left(\sum_{k=1}^{n} W_k\right)^{-1}$$

and $W_1, \cdots, W_n$ are independent, exponentially distributed with parameter one (Breiman [1, Chap. 13], Karlin and Taylor [9, Chap. 13]). Using the Central Limit Theorem for (moderately) large deviations (Feller [3, Chap. 16]), we have ($E(W_j) = \operatorname{var}(W_j) = 1$):

$$\Pr\left(\left|\sum_{j=1}^{n} W_j - n\right| \geqq n^{1/7} \cdot n^{1/2}\right) = O\left(\exp\left(-\frac{n^{2/7}}{2}\right)\right).$$

So,

$$\Pr(M_n \leqq x) \leqq \Pr\left(\max_{1 \leqq j \leqq n} W_j \leqq x(n+n^{9/14})\right) + O\left(\exp\left(-\frac{n^{2/7}}{2}\right)\right).$$

Here,

$$\begin{aligned}\Pr\left(\max_{1 \leqq j \leqq n} W_j \leqq x(n+n^{9/14})\right) &= (1-\exp(-x(n+n^{9/14})))^n \\ &\leqq \exp(-n \exp(-x(n+n^{9/14}))) \\ &= \exp(-\exp(\ln n - (\ln n - \ln\ln n - \rho)(1+n^{-5/14}))) \\ &\leqq \exp(-\exp(\ln\ln n + \rho')) = n^{-e^{\rho'}} \quad \forall \rho' \in (0, \rho). \qquad \square\end{aligned}$$

(3) It remains to show that $nU_n \rightarrow 2$ in probability, where $U_n = \sum_{j=1}^n L_j^2$. To this end, we note that $U_n$ coincides, in distribution, with

$$\frac{(\sum_{j=1}^n W_j^2)}{(\sum_{k=1}^n W_k)^2},$$

and, by the weak law of large numbers,

$$(\text{P}) \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{j=1}^{n} W_j^2 = \int_0^\infty x^2 e^{-x}\, dx = 2,$$

$$(\text{P}) \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{j=1}^{n} W_j = \int_0^\infty x e^{-x}\, dx = 1;$$

so,

$$(\text{P}) \lim_{n \rightarrow \infty} nU_n = 2.$$

**Acknowledgment.** I am very grateful to Don Knuth for inviting me to spend the year of 1988 at Stanford. Don introduced me to the stable marriage problem and, in particular, suggested that I undertake an asymptotic study of the expected stable matchings. In an early stage of this work, I was greatly helped by insightful comments on the problem made by N. G. de Bruijn in a letter to Don. Many discussions of the work in progress with Don, and also Andy Goldberg, Vlad Lifschitz, and Herb Wilf, invariably encouraged me to keep going. Don read the first draft of the paper and made numerous suggestions that were extremely useful to me in writing the final version. My sincere thanks to David Aldous, Brad Efron, Don Iglehart, and Sam Karlin for opportunities to present a talk on the stable matchings problem.

## REFERENCES

[1] L. Breiman, *Probability*, Addison-Wesley, Reading, MA, 1968.

[2] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Ann. Math. Statist., 23 (1952), pp. 493–509.

[3] W. Feller, *An Introduction to Probability Theory and Its Applications*, 2nd ed., John Wiley, New York, 1971.

[4] D. Gale and L. S. Shapley, *College admissions and the stability of marriage*, Amer. Math. Monthly, 69 (1962), pp. 9–15.

[5] D. GUSFIELD, R. IRVING, AND P. LEATHER, *Every finite distributive lattice is a set of stable matchings for a small stable marriage instance*, J. Combin. Theory Ser. A, 44 (1987), pp. 304–309.

[6] R. IRVING AND P. LEATHER, *The complexity of counting stable marriages*, SIAM J. Comput., 15 (1986), pp. 655–667.

[7] R. IRVING, *An efficient algorithm for the "stable roommates" problem*, J. Algorithms, 6 (1985), pp. 577–595.

[8] R. IRVING, P. LEATHER, AND D. GUSFIELD, *An efficient algorithm for the "optimal" stable marriage*, J. Assoc. Comput. Mach., 34 (1987), pp. 532–543.

[9] S. KARLIN AND H. M. TAYLOR, *A Second Course in Stochastic Processes*, Academic Press, New York, 1981.

[10] D. E. KNUTH, *Mariages Stables et leurs relations avec d'autres problèmes combinatoires*, Les Presses de l'Université de Montréal, Montréal, 1976.

[11] D. G. MCVITIE AND L. B. WILSON, *The stable marriage problem*, Comm. ACM, 14 (1971), pp. 486–490.

[12] A. RÉNYI, *Probability Theory*, North-Holland, New York, 1970.

[13] L. B. WILSON, *An analysis of the stable marriage assignment problem*, BIT, 12 (1972), pp. 569–575.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 550-581, November 1989

# A MATHEMATICAL MODEL FOR PERIODIC SCHEDULING PROBLEMS*

PAOLO SERAFINI† AND WALTER UKOVICH‡

**Abstract.** A mathematical model is proposed for scheduling activities of periodic type. First a model is proposed for scheduling periodic events with particular time constraints. This problem, which could be considered the extension to periodic phenomena of ordinary scheduling with precedence constraints, is shown to be NP-complete. An algorithm for it of implicit enumeration type is designed based on network flow results, and its average complexity is discussed. Some extensions of the model are considered. The results of this first part serve as a basis in modelling periodic activities using resources. Several cases are considered. Finally some applications are presented for which the proposed model can be a useful tool.

**Key words.** scheduling, periodic scheduling, cyclic scheduling

**AMS(MOS) subject classifications.** 90B35, 90C35

## 1. Introduction.

### 1.1. Scope of the paper.

This paper deals with the problem of scheduling periodic phenomena, that is, events and activities to be identically repeated at a constant rate. Periodic phenomena may arise either naturally, or as the consequence of imposed constraints for reasons of convenience or efficiency. In particular this may occur whenever a finite set of actions must be repeated with an infinite time horizon.

Examples of periodic phenomena are quite numerous and range over a wide spectrum of different applications involving problems of production, maintenance, transportation, vehicle scheduling, personnel scheduling, control, real-time processing, and so on.

The aim of this paper is to develop a consistent model into which some of the above problems might be conveniently approached both from the point of view of model clarity and from that of computational effort. We do not pretend to provide efficient and convenient methods to solve every kind of periodic scheduling problem, even if we think that the model proposed in this paper is sufficiently general to encompass several practical problems.

We realize that generality can be a drawback from the point of view of computational efficiency, which may be typically improved by avoiding generality and exploiting particular structures. Hence the computational approach we propose, while exhibiting a good average behaviour, should nevertheless be regarded as a departure point for the development of more sophisticated "ad hoc" exact and/or heuristic algorithms. In this sense, even if part of the paper is algorithmically oriented, its main concern is more about modelling issues.

A convenient way of defining and explaining the scope and the characteristics of our model consists in comparing it to classical (nonperiodic) "project scheduling" problems like the ones usually dealt with by CPM/PERT techniques. The ordinary ingredients of ordinary project scheduling are activities and their precedence constraints. The classical problem then consists in finding a feasible activity schedule minimizing the overall project completion time. This may be accomplished in polynomial time by dynamic programming algorithms working on an acyclic directed graph whose structure expresses the activities (nodes) and their precedence constraints (arcs).

* Received by the editors June 3, 1987; accepted for publication (in revised form) November 29, 1988.

† University of Udine, Dipartimento dí Matematica e Informatica, Via Zanon 6, 33100 Udine, Italy.

‡ University of Trieste, Dipartimento dí Elettrotecnica, Elettronica, e Informatica, Via Valerio 10, 34127 Trieste, Italy.

In the periodic version we are considering we deal with periodic activities with a given common period; the precedence constraints, which are meaningless in a periodic setting, are reformulated as "time window" constraints affecting the relative position of pairs of activities within the period.

Clearly there is no completion time to "minimize," and the period value, in the case of nonnatural periodicity, is a design parameter. We approach the problem of periodic scheduling by considering the period as a parameter given a priori. So we are primarily interested in finding feasible schedules with respect to the time window constraints and do not consider the problem of the best choice of the period value. If particular problems do exhibit natural objectives to minimize, then the model proposed here could be regarded as an initial formulation to be further refined.

A major difference between ordinary and periodic scheduling is given by their computational complexity. In fact, while ordinary project scheduling is polynomial, periodic scheduling is NP-complete, even if no resources are involved. On the other hand, the presence of resources makes ordinary project scheduling a qualitatively much harder problem, since it becomes NP-hard in most cases. Quite differently, the introduction of resources in periodic scheduling makes the problem only quantitatively more difficult, leaving it in the NP class. We remark that, in presence of resources, the same type of computational complexity appears both in ordinary and periodic scheduling. Therefore we think that our model could also benefit from the research carried out in ordinary scheduling in order to gain computational improvements.

**1.2. A brief review of pertinent results.** Despite its potential interest, papers on periodic scheduling constitute only a negligible fraction of the overall literature on scheduling problems (compare, for instance, [10], [17], [22]). Most of them are concerned with specific applications, such as cyclic staffing, production planning, etc. Both the models and the methods proposed for such cases are often strongly application-oriented, so they may bring little insight for a general approach.

A formulation of periodic scheduling problems with several similarities to our model is proposed by Dauscha, Modrow, and Neumann [8]. They deal with activities of fixed duration subject to constraints on the minimum time distance between their ordered activation times, thus giving rise to "cyclic precedence orders." As a mathematical tool they use the "cyclic sequence types" expressing the number of periods needed to carry out any given cyclic set of activities in a given order. Then they derive some interesting properties of the cyclic sequence types and a necessary and sufficient condition on them that guarantees the existence of a periodic schedule. A method is presented for constructing a periodic schedule, once a feasible cyclic sequence type is provided with time complexity bounded by the third power of the number of constraints; moreover, the overall problem of finding an admissible cyclic schedule is shown to be NP-complete and is formulated as a mixed-integer programming problem with Boolean variables. An application to a simple problem concerning the setting of a traffic light system operating periodically is thoroughly carried out.

Other aspects of periodic scheduling problems are considered by Kats [15]. He is concerned with finding periodic scheduling and routes for a system of interprocessor conveyors serving a production line. Processing and interprocessor transfer times are bounded to lie within prescribed intervals, and a minimum period schedule is sought. A result that allows the number of operators necessary along one route to be expressed as the number of periods necessary to travel through it is used to restrict the search for the optimal schedule to the solutions minimizing the number of operators in a given period. A branch-and-bound solution method is proposed, acting on the elements of the matrix expressing the minimal number of periods required by a conveyor to perform two successive movements for a given schedule. It solves in polynomial time the case with fixed operation lengths, which corresponds to a resource minimization in our formulation.

A similar approach is considered by Orlin [18], who minimizes the number of vehicles to meet a fixed periodic schedule, with deadheading allowed. He formulates the problem by using a periodic version of the partially ordered sets and solves it in a polynomial time using a periodic version of the Dilworth's theorem, leading to results similar to the ones found in the next § 3.2, i.e., resource minimization in presence of a fixed activity schedule. The same problem may be reformulated as a coloring problem for a "periodic interval graph." A similar coloring problem, concerning "circular arc graphs" (see [28], [11]), corresponds again to the problem of assigning vehicles to a fixed periodic schedule, with the additional requirement that the same vehicle must always perform the same activity. As Orlin points out, although in general the latter problem is NP-hard, some simpler formulations may be solved in polynomial time as in Orlin, Bonuccelli, and Bovet [19].

Again similar problems are investigated by Gertsbakh and Gurevich [12], [13] in the framework of vehicle scheduling for a given time table by using the concept of "deficit function." Although their approach is entirely different from the ones previously outlined and from ours as well, the results they obtain are quite similar, as far as resource minimization is concerned. Along this line of research we may also quote a paper by Ceder and Stern [7].

An interesting class of periodic scheduling problems arising from production planning is the "economic lot scheduling problem." Its objective is finding an optimal schedule that allows cyclic production patterns for several products that are made on a single machine, in such a way that setup and inventory costs per unit time are minimized. A peculiar aspect of this problem is that different products may well have different frequencies of production, i.e., production periods of different lengths. Vemuganti [29] proposes a mixed-integer linear programming approach for this problem. Hsu [14] proves that this problem is NP-hard and presents an implicit enumeration scheme to test the feasibility of a given set of production periods. Problems with similar structure may also be approached by our model according to the results of §§ 2.5 and 3.4.

A similar class of problems, in which different periodic events may have different recurrence rates, is dealt with by Burkard [6]. He provides some interesting results for several cases, dealing with two or more events, with respect to different types of objective.

Another problem concerning periodic activities with different period lengths is considered by Park and Yun [21]. They seek to minimize the maximum amount of resources required by such activities. The problem is formulated as a 0-1 linear programming problem and the Chinese Remainder Theorem is used to decompose it.

A quite different kind of periodic scheduling problem concerns personnel scheduling. Here time is normally discretized in large intervals and the aim is usually to meet some given periodic workload pattern with weekly periodicity while trying to satisfy some additional requirements. Such problems originated a relatively large number of papers; just to quote a few of them, see the early paper by Baker [1] and the ones by Bartholdi, Orlin, and Ratliff [3], Bartholdi [2], Bechtold [5], Emmons [9]. Despite the interest of these types of problems and some apparent similarities with the previous problems, they are rather different and so we do not provide models for them, although it is perhaps possible to adapt our model to some of them.

An overall glance at the literature dealing with periodic scheduling reveals a wide spectrum of applications with a relatively more restricted range of approaches proposed to solve them: usually NP-completeness is shown for the general problem and an integer (or mixed-integer) LP formulation is presented; then combinatorial heuristics for it or polynomial algorithms for simpler subproblems are proposed. Surprisingly enough, papers referring to different applications never consider the literature relative to other applications, even if they deal with abstract models that could be conveniently applied or extended to a larger class of problems. With this perspective, the present paper's objective is to provide a first unifying framework through abstract models apt to formalize basic features and phenomena that may be faced in several different applications.

**1.3. Plan of the paper.** The paper is structured in three main parts. In the first part (§ 2) the periodic event scheduling problem is defined with reference to time constraints only. Then in the second part (§ 3) resource constraints are taken into account, leading to more complex problems. In the third part (§ 4) some applications are described to which the previous models can be applied.

The detailed plan of the paper is as follows. The periodic event scheduling problem is first defined in terms of the abstract concept of a periodic event, its schedule, and a particular type of constraints for the schedule that affect the relative positions of pairs of periodic events. Then it is shown that the resulting problem is NP-complete.

The problem is reformulated as a network problem and an algorithm is derived from this formulation based on an implicit enumeration technique. The probabilistic performance of the algorithm is discussed and some computational evidence is provided.

Then the initial model is embedded with additional features, such as multiple periodicities and sequencing constraints, which turn out to be convenient when dealing with resources and therefore in several applications. At the conclusion of this part, a relationship with the triangular Traveling Salesman Problem is pointed out.

In the second part, activities and resources in a periodic setting are first defined and the peculiar structure of resource scheduling is investigated. As a consequence, the problem of scheduling periodic activities with resource constraints is reduced to the models developed in the previous part. In particular, the problem of minimizing the number of resource units is easily solved through a weighted assignment problem (obtaining a result similar to Bartlett [4], Orlin [18] and Gertsbakh and Gurevich [12], [13]), whereas different models of increasing complexity are presented for the problem of scheduling the activities with different types of resource constraints.

The last part is devoted to three applications. The first is a periodic version of the Job-Shop Problem. Similarities and dissimilarities between the periodic and the non-periodic version are investigated. A second application concerns the problem of vehicle scheduling according to a periodic time table. Finally, the problem of traffic light scheduling is outlined and a simplified version of it is modelled according to the previous results.

This paper is the refinement and extension of the preliminary results that already appeared in [24].

## 2. The periodic event scheduling problem.

**2.1. Basic definitions.** A *periodic event* $\varepsilon$ is a countably infinite set of events $e(p)$, indexed by $p \in Z$, with occurrence times $t(e(p)) \in R$, such that for each $p$, $t(e(p)) - t(e(p-1)) = T$ ($Z$ denotes the set of integers and $R$ the set of reals). $T$ is referred to as the *period* and $e(p)$ as the $p$th occurrence of the periodic event $\varepsilon$. We also define $t(\varepsilon) := t(e(p)) \bmod T$.

A periodic event $\varepsilon$ is said to be *scheduled* if an element $\tau(\varepsilon) \in \Phi := R/T$ is associated to it, such that $\tau(\varepsilon) = \Pi \cdot t(e(p))$, for all $p$, where $\Pi$ is the canonical projection from $R$ to the abelian group $\Phi$.

Let $D = [d^-, d^+]$ be any real closed interval. Then a *span* $\Delta$ of $\Phi$ is defined by $\Delta := \Pi \cdot D$. Note that $\Phi$ itself is a span by this definition; *proper spans* are defined by excluding the cases $\Delta = \Phi$ and $\Delta = \varnothing$.

Any proper span may be denoted by $\Delta = [\delta^-, \delta^+]$, with $\delta^- = \Pi \cdot d^-$ and $\delta^+ = \Pi \cdot d^+$, or, with abuse of notation, by $\Delta = [d^-, d^+]$, or even by $\Delta = [d^-, d^+]_T$ if it is necessary to specify the period. Note that $\Delta = [d^- + zT, d^+ + zT]_T$ for all $z \in Z$. The ambiguous case in which $\delta^- = \delta^+ = \delta$ will be proposed as $[\delta^-, \delta^+] = \{\delta\} = \Pi \cdot [d, d]$.

A *span constraint a* is a relationship involving an ordered pair of periodic events $(\varepsilon^-(a), \varepsilon^+(a))$ and a proper span $\Delta(a)$ and requiring that $\tau(\varepsilon^+(a)) - \tau(\varepsilon^-(a)) \in \Delta(a)$.

Loosely speaking, while nonperiodic events are arranged in a linear chronological order, periodic events have a cyclic chronological order and their occurrence times $\tau$ may be considered as points on a circumference which is scanned e.g., clockwise. A span constraint between two periodic events restricts the occurrence time of one event relative to the occurrence time of the other event to belonging to a certain time window on the circumference.

Throughout the paper we have tried as much as possible to employ a notation which should put in evidence the type of concepts with which we are dealing. So we use Greek letters ($\varepsilon$, $\tau$, $\Delta$, $\delta$, etc.) to denote periodic or group quantities, and Latin letters ($e$, $t$, $D$, $d$, etc.) to denote either nonperiodic quantities or real representations of group quantities.

The Periodic Event Scheduling Problem (PESP) is the following one.

Given
— a finite set $N$ of $n$ periodic events with a common period $T$;
— a finite set $A$ of span constraints;
find a *schedule* $\tau(\varepsilon) \in \Phi$, for all $\varepsilon \in N$, satisfying all span constraints.

For an example of a simple PESP instance refer to Fig. 1.1.

Multiple span constraints for the same pair of periodic events can also be imposed, thus allowing us to consider the intersection of the corresponding spans. However, a remarkable fact, due to the periodic structure, is that the intersection of two spans is not necessarily a span: in fact, it can be the union of two disjoint spans. As an example consider $[4, 5]_{10} \cup [7, 12]_{10} = [7, 15]_{10} \cap [4, 12]_{10}$ (see Fig. 1.2). Hence it is possible to model dichotomy constraints of the type $\tau(\varepsilon_2) - \tau(\varepsilon_1) \in \Delta_1 \cup \Delta_2$ by simply imposing

| | | |
|---|---|---|
| $T = 10$ | $N = \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$ | $A = \{a_1, a_2, a_3\}$ |
| $\varepsilon^-(a_1) = \varepsilon_1$, | $\varepsilon^+(a_1) = \varepsilon_2$, | $\Delta(a_1) = [3, 6]$ |
| $\varepsilon^-(a_2) = \varepsilon_2$, | $\varepsilon^+(a_2) = \varepsilon_3$, | $\Delta(a_2) = [2, 4]$ |
| $\varepsilon^-(a_3) = \varepsilon_1$, | $\varepsilon^+(a_3) = \varepsilon_3$, | $\Delta(a_3) = [0, 5]$ |

The two feasible schedules are shown below:

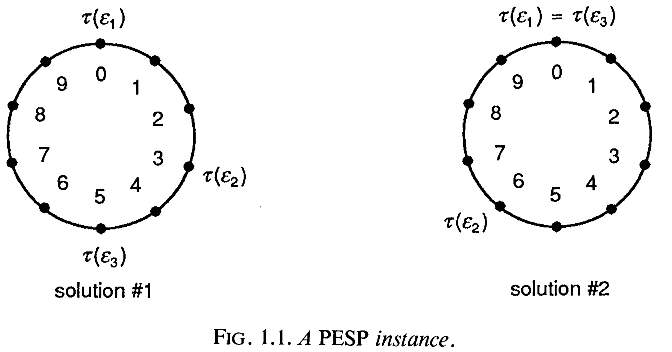


FIG. 1.1. *A* PESP *instance.*

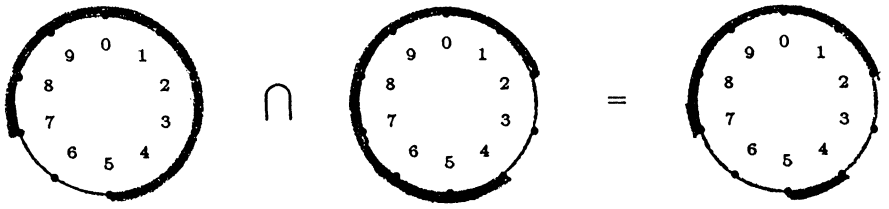


FIG. 1.2. *Intersection of spans.*

two appropriate span constraints for the same pair of periodic events. Of course, constraints involving the union of more than two spans can also be dealt with in a similar way.

**2.2. Properties of PESP.**

THEOREM 1. PESP *is* NP-*complete.*

*Proof.* It is easy to see that PESP is in NP. In order to prove NP-completeness, a polynomial transformation from the Hamiltonian Circuit Problem to PESP is produced. Let $G = (V, E)$ be a nondirected graph with $n$ vertices for which a Hamiltonian circuit is sought. Let $E'$ and $F'$ be the arc sets of $G$ and of the complete graph $K_n$, respectively, with arcs arbitrarily oriented. Then consider the following instance of PESP with $T = n$, $N$ identified with $V$, $A$ identified with $F'$, $\Delta(a) = [1, n-1]$ if $a \in E'$ and $\Delta(a) = [2, n-2]$ otherwise.

Then PESP has a solution if and only if $G$ has a Hamiltonian circuit. In fact, suppose a solution $\tau(\varepsilon)$ for PESP is produced: then $t(\varepsilon') - t(\varepsilon'') \bmod n \geqq 1$ for all $\varepsilon', \varepsilon'' \in N$, $\varepsilon' \neq \varepsilon''$, and there exists a cyclic permutation $P$ on $N$ such that $t(P(\varepsilon)) - t(\varepsilon) \bmod n = 1$ with $\varepsilon$ and $P(\varepsilon)$, adjacent vertices in $G$.

Conversely, suppose that a cyclic permutation $P$ is generated corresponding to a Hamiltonian circuit of $G$; it suffices to set $\tau(\varepsilon_1) = 0$ and $\tau(P(\varepsilon_i)) = \tau(\varepsilon_i) + 1$ in order to get a solution for PESP. □

It is immediate to check that PESP is also strongly NP-complete (see [20]).

It is useful to give a network representation of PESP. Given any instance of PESP, consider $G = (N, A)$ as a network: periodic events are considered as nodes and span constraints are considered as directed arcs. Furthermore, the following set of real spans is associated to each arc $a$

$$D(a, z) := [d^-(a) + zT,\ d^+(a) + zT] \quad \forall z \in Z$$

in such a way that $\Pi \cdot D(a, z) = \Delta(a)$.

For any network, a function $u : N \to R$ is called a *potential*, and the corresponding function $v : A \to R$ such that $v(a) = u(\varepsilon^+(a)) - u(\varepsilon^-(a))$ is called a *tension* (see [23]). In the context of the network representation of PESP, a span constraint is a constraint involving the tension of some arc $a$ so that $v(a) \in D(a, z)$ for some integer $z$.

Then PESP is equivalent to the following network problem.

Find a potential $u$ such that $v(a) \in D(a, z(a))$ for some $z(a) \in Z$, for all $a \in A$.

Figure 1.3 gives a network representation of the PESP instance of Fig. 1.1 and of its solution #1. Of course $\tau(\varepsilon) = \Pi \cdot u(\varepsilon)$.

If the resulting network turns out to be disconnected, then obviously the instance splits into a certain number of instances (one for each connected component) which can be solved independently. In view of this remark we shall always assume that the network is connected.

It is important to note that the above network problem reduces to the *Feasible Differential Problem* (FDP) if the values of the $z$'s for all arcs are fixed a priori (see [23]). This fact will be exploited in designing an algorithm for PESP.

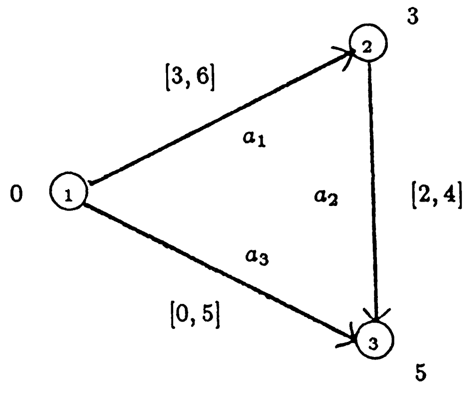


FIG. 1.3. *Network representation of* PESP.

The FDP is polynomial with complexity $O(n^3)$ and the difficulty for PESP comes in by allowing $z$ to assume any integer value. It is interesting to point out some other differences between FDP and PESP.

For any circuit $C$, let $C^+$ and $C^-$ be the set of arcs having the same and the opposite orientation, respectively, of the circuit $C$. Then define:

$$D(C):=\left[\sum_{a\in C^+} d^-(a)-\sum_{a\in C^-} d^+(a),\ \sum_{a\in C^+} d^+(a)-\sum_{a\in C^-} d^-(a)\right].$$

For instance, consider in Fig. 1.3 the unique circuit $C$ oriented clockwise. Then $C^+ = \{a_1, a_2\}$, $C^- = \{a_3\}$ and $D(C) = [3+2-5, 6+4-0] = [0, 10]$.

An obvious necessary condition for PESP to have a solution is the following:

THEOREM 2. PESP *has a solution only if for any circuit $C$ there exists $z(C) \in Z: z(C)T \in D(C)$.*

*Proof.* First note that there is no difference in reversing the orientation of some arc $a$ and replacing its span $[d^-(a), d^+(a)]$ with $[-d^+(a), -d^-(a)]$. Now take any circuit $C$ and reverse the orientation of the arcs in $C^-$. Given a feasible potential, the sum of the tensions along the circuit $C$, computed according to the new orientations, must be zero. Since $d^-(a) \leqq v(a) - z(a)T \leqq d^+(a)$ if $a \in C^+$ and $-d^+(a) \leqq v(a) + z(a)T \leqq -d^-(a)$ otherwise, the thesis follows by simply summing up the inequalities along the circuit. □

In the example of Fig. 1, this condition is satisfied with both $z = 0$ and $z = 1$. However, the condition is not sufficient. Consider the instance in Fig. 2 that satisfies the condition but is not feasible. This result of nonsufficiency was also expected by theoretical

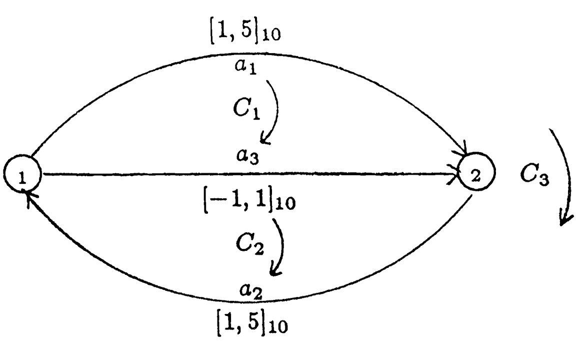


FIG. 2

considerations: in fact, the sufficiency of the above condition would allow a concise certificate, namely a circuit, for the complement of the Hamiltonian circuit problem. Note the difference with respect to the FDP where a similar condition, namely $0 \in D(C)$, is both necessary and sufficient (see [23 p. 193]).

**2.3. An algorithm for PESP.** In the following we shall borrow the terminology from [23, Chap. 6] and some basic techniques used by the algorithm. We shall also use the following notation: let $P(J, z_K)$ denote a new problem obtained from the network representation of PESP by restricting the span constraints to a subset $J \subset A$, and by fixing the values $z(a)$ for the spans relative to the arcs $a$ of a subset $K \subset J$ to an array $z_K$ of integers. By this notation PESP $= P(A, z_\varnothing)$.

The algorithm for PESP is outlined in Fig. 3. Below we describe its features in detail. An instance of PESP is displayed in Fig. 4.1 and the description of the algorithm will also refer to Fig. 4 for ease of presentation.

First let us note that it is immediate to find a solution for PESP if the network is a tree: it is just a matter of assigning an arbitrary potential to an arbitrary node and then assigning recursively the potentials to the other nodes so that the tensions on the branches of the tree are feasible (see Fig. 4.2).

Indeed, the algorithm is based on the idea of first solving a problem $P(B, z_\varnothing)$, with $B$ the arc set of a spanning tree $(N, B)$ of $G$ and then trying to make the span constraints relative to the chords of the tree feasible one by one. In order to accomplish the latter task, we shall design an implicit enumeration procedure for the integers $z$ relative to the spans of the chords.

It is important to note that the integers $z$ relative to the span constraints of $B$ can be fixed to any value without altering feasibility or infeasibility of the PESP instance at hand, or, more formally, $P(A, z_\varnothing)$ is feasible if and only if $P(A, z_B)$ is feasible for any fixed $z_B$. In fact, suppose that there exists a feasible potential and that the integer $z(\bar{a})$, with $\bar{a} \in B$, has the feasible value $z_1$; let $(N^+, B^+)$ and $(N^-, B^-)$ be the trees obtained from $(N, B)$ by removing the arc $\bar{a}$, with $\varepsilon^+(\bar{a}) \in N^+$ and $\varepsilon^-(\bar{a}) \in N^-$. If we replace $z_1$ with $z_2$ a new feasible potential is immediately obtained by adding $(z_2 - z_1)T$ to all potentials on $N^+$.

We call *tree integrality* the property for a given spanning tree $(N, B)$ to admit arbitrary fixed values for the integers $z(a)$, for all $a \in B$ without affecting the feasibility of the PESP instance at hand. It is rather obvious that tree integrality holds for any spanning tree of a PESP instance and so this definition might seem superfluous. However, we shall extend the PESP model in § 2.5 and in this extended model tree integrality will not always hold.

The initialization of the algorithm consists in finding a minimal spanning tree $(N, B)$ with respect to the costs $(d^+(a) - d^-(a))$, sorting the chords as $c_1, c_2, \cdots, c_q$ in order of increasing costs $(d^+(c_i) - d^-(c_i))$ and assigning an initial potential $u$ so that $v(a) = (d^+(a) + d^-(a))/2$, for all $a \in B$. By this choice we have implicitly set $z(a) = 0$, for all $a \in B$. We shall explain in § 2.4 why $B$ has to be a *minimal* spanning tree, the chords have to be *sorted* and the tension is initially set in the *middle* of the span.

The main body of the algorithm is made up of the implicit enumeration procedure to make the chord span constraints feasible one by one. With every PESP instance, a search tree is associated with at most $q$ levels, corresponding to the $q$ chords (obviously $q = |A| - |N| + 1$ is also the number of independent circuits of $G$). The root of the search tree at level zero corresponds to problem $P(B, z_B)$. All other nodes correspond to some problem $P(J, z_J)$ with $B \subset J \subset A$; at the $i$th level $J = B \cup c_1 \cup \cdots \cup c_i$. In order to explain the way the successors of a given node are generated in the next level of the search tree we shall consider in detail how the successors of the root node are generated by referring explicitly to Figs. 4.1–4.7.

The successors of Problem $P(B, z_B)$, i.e., the root node, are the Problems $P(B \cup c_1, z_{B \cup c_1})$, differing from each other for the value assigned to $z(c_1)$. The set of admissible values for $z(c_1)$ is generated in the following way (see Fig. 4.3): the tension $v(c_1)$ induced by the feasible potential for Problem $P(B, z_B)$ is equal to 0.5 and it is not feasible for Problem $P(B \cup c_1, z_B)$ since $v(c_1) \notin D(c_1, z)$, for all $z$.

In order to make Problem $P(B \cup c_1, z_B)$ feasible we have two possibilities, either to raise the tension by at least the quantity $v^+ = 2.5$ so that it belongs to the span at right with value $z(c_1) = 0$ or to lower it by at least the quantity $v^- = 4.5$ so that $v(c_1) \in D(c_1, -1)$. The problem of raising (lowering) the tension on $c_1$ is the dual problem of a shortest path problem from $\varepsilon^-(c_1)$ to $\varepsilon^+(c_1)$ (from $\varepsilon^+(c_1)$ to $\varepsilon^-(c_1)$) on the network $(N, B)$ with respect to the spans $[d^-(a) + z_a T - v(a), d^+(a) + z_a T - v(a)]$ for $a \in B$ (again see [23]), or alternatively, the dual problem of a shortest path problem on a directed graph $(N, B')$ with $B'$ obtained from $B$ by replacing each arc $a \in B$ with a pair of directed arcs $\varepsilon^-(a) \to \varepsilon^+(a)$ and $\varepsilon^+(a) \to \varepsilon^-(a)$ with lengths $d^+(a) + z_a T - v(a)$ and $v(a) - d^-(a) - z_a T$, respectively. Roughly speaking, these lengths represent the maximum amounts that the tension on the arc $a$ can be "stretched" or "shrunk," respectively. The directed graph $(N, B')$ is shown in Fig. 4.4 (the lengths of each pair of arcs are equal because of the initial choice of the tension in the middle of the spans; this situation does not happen, in general of course, on the other nodes of the search tree).

The tension can be raised up to the span $D(c_1, 0)$ if and only if the shortest path value from $\varepsilon^-(c_1)$ to $\varepsilon^+(c_1)$ is at least $v^+$ and the tension can be lowered up to the span $D(c_1, -1)$ if and only if the shortest path value from $\varepsilon^+(c_1)$ to $\varepsilon^-(c_1)$ is at least $v^-$. If so, the new feasible potential is computed in the following way (let us illustrate it for the span $D(c_1, 0)$, the other case is symmetric): Dijkstra's algorithm generates a tree of shortest paths from $\varepsilon^-(c_1)$ to all other nodes by appending recursively the nearest node to an expanding tree starting from the source $\varepsilon^-(c_1)$. Let $w(\varepsilon)$ be the shortest path value from $\varepsilon^-(c_1)$ to a generic node $\varepsilon$. Then, in order to compute the new feasible potential it is not necessary to run Dijkstra's algorithm to its full completion; it is enough to stop it as soon as a node $\bar{\varepsilon}$ is reached with distance $w(\bar{\varepsilon}) \geqq v^+$. Let $N'$ be the set of nodes reached by Dijkstra's algorithm up to this point with the exclusion of the last node $\bar{\varepsilon}$. Then the potential is updated according to

$$u(\varepsilon) := \begin{cases} u(\varepsilon) + w(\varepsilon) - v^+ & \text{if } \varepsilon \in N' \\ u(\varepsilon) & \text{otherwise.} \end{cases}$$

In the example of Figs. 4.1–4.7, both raising and lowering the tension of the stated amounts are possible, so both $z(c_1) = -1$ and $z(c_1) = 0$ correspond to admissible successors of $P(B, z_B)$. Other admissible successors can be generated by raising (lowering) the tension even more up to the span next to the right (left), i.e., $D(c_1, 1)$ ($D(c_1, -2)$). In this case it is required that the shortest path value be at least 12.5 (14.5). For this example no feasible tension can be established for the spans $D(c_1, 1)$ and $D(c_1, -2)$ and this implies in turn that no feasible tension can exist for all spans $D(c_1, z)$ with $z \geqq 1$ or $z \leqq -2$.

So the successors of $P(B, z_B)$ are all the problems $P(B \cup c_1, z_{B \cup c_1})$ with values $z(c_1)$ giving raise to feasible potentials (these $z(c_1)$ constitute a set of "adjacent" integers), plus two infeasible problems corresponding to the integers immediately at the left and at the right of the set of feasible integers. These last two problems must belong to the search tree because they are necessary for the algorithm to backtrack.

The above discussion for the successors of the root node of the search tree is also valid for the successors of any node $P(J, z_J)$. So the structure of the search tree is the following: each feasible $P(J, z_J)$ has at least two successor problems and exactly two of them are nonfeasible. Nonfeasible problems do not have successors. Feasible solutions of the PESP problem are found as the feasible potentials for the problems at the $q$th level (the bottom) of the search tree. See in Fig. 4.5 the search tree of the example. Note that a infeasible instance of PESP may have less than $q$ levels.

It is important to note that whenever the tension $v(c)$ on a certain chord $c$ cannot be made feasible with respect to a certain span $D(c, z)$, the shortest path between the extremes of the chord together with the chord itself form a circuit for which the span constraints are too tight to accommodate a feasible potential. Let us call this circuit a *blocking circuit*, and the other chords besides $c$ possibly belonging to the blocking circuit *blocking chords*. We shall use the phrase *blocking gap* to indicate the quantity by which $v(c)$ differs from the span to which it is modified (i.e., the blocking gap is either $d^{-}(c) + z_i T - \max v(c)$ or $\min v(c) - d^{+}(c) - z_i T$). As we shall see, the information provided by a blocking circuit is very valuable.

The algorithm we suggest performs a depth-first visit of the search tree by selecting at each level the most central successor in the next level, and backtracking to the most central nonvisited node of the previous level whenever the two infeasible nodes of the level are visited. See in Fig. 4.6 the nodes of the search tree actually visited. Note that, in view of the previous considerations, a visit of a node consists of a run of the Dijkstra's algorithm.

The performance of this "naive" algorithm can be improved if we introduce a device, which may be called *accelerated backtracking*, based on the following considerations: when the algorithm backtracks from a certain level, this is because a certain set of blocking circuits has been discovered at deeper levels. In order to find a feasible potential the only integers $z$ of the previous levels which should be changed are the ones relative to the blocking chords. Therefore there is no need to guess other $z$ values at the level immediately above if this refers to a chord nonpresent in any blocking circuit. Thus the accelerated backtracking device consists in backtracking until a level is found corresponding to a blocking chord.

Figure 4.7 shows the nodes of the search tree of the example actually visited by using the accelerated backtracking.

In order to appreciate the importance of the accelerated backtracking let us consider a PESP instance whose network consists of two networks $G_1$ and $G_2$ linked through a node. The instance is actually made up of two independent instances, but let us suppose that we have not realized the existence of this simpler structure and solve the problem globally with the "naive" algorithm. Let us also suppose that the problem is infeasible on $G_2$ and feasible on $G_1$ and that the chords are sorted so that all chords in $G_1$ correspond to the first levels of the search tree. So the algorithm will first find a feasible potential with respect to $G_1$ and then will discover that a feasible potential for the whole network cannot be found with the fixed $z$ values in $G_1$. Therefore the "naive" algorithm will backtrack to guess other $z$ values for the chords in $G_1$. But this is actually a waste of time since these values do not affect feasibility at all on $G_2$. This fact can be discovered by looking at the blocking chords. In fact all blocking circuits in $G_2$ are necessarily confined in $G_2$, so when backtracking from $G_2$ to $G_1$, the algorithm may realize that there are no blocking chords in $G_1$, thus making a unique backtracking up to the root node and concluding infeasibility of the instance.

The algorithm can also be embedded with a routine which records some of the blocking circuits found as well as the relative gaps. This kind of information turns out to be useful, in case of infeasibility, if the possibility is considered of relaxing some span constraints in order to find a feasible solution.

**2.4. Considerations for the probabilistic analysis of the algorithm.** With regard to the worst-case computational complexity of the algorithm described in the previous section, we may note that it is given by $O(s(n)n^2)$, with $O(n^2)$ being the complexity relative to the visit of a node (running the Dijkstra's algorithm) and $s(n)$ being a bound on the number of nodes of the search tree. Obviously $s(n)$ is exponential in the worst case. Nevertheless it is possible to arrange things so that at least the probabilistic behaviour is acceptable.

Of course the less the nodes of the search tree are the faster the computation is. It turns out that the number of nodes not only depends on the instance data, but also on the way the data are "fed" to the algorithm. Intuitively, it is better to have the most "prolific" nodes at the bottom rather than at the top of the search tree, and this situation can be achieved if the arcs considered first are the ones with narrowest spans. This explains why in the stated algorithm we choose $B$ as a minimal spanning tree and the chords are sorted. Moreover, the initial value for the tensions is chosen in the middle of the span in the presumption that selecting the most central successor of a given node of the search tree gives the highest probability of finding a feasible successor.

These considerations are rather informal. However, it is possible to support them with more rigorous arguments. In fact, each feasibility problem $P(J, z_J)$ is a relaxation of $P(A, z_J)$ and we may think of it as an optimization problem with objective function $\sum_{a \in J}$ distance $(v(a), D(a, z_a))$. Therefore a node of the search tree is nonfeasible if and only if the optimal value of its corresponding relaxation is strictly positive. Averaging on a random set of instances the probability for a node to be nonfeasible is monotonically decreasing with respect to the span lengths of the arcs in $J$ and increasing with respect to $|J|$. The same conclusion can be reached if we consider that nonfeasibility of a node means that a shortest path between two given nodes is shorter than the quantity by which $v(a)$ "misses" the real span $D(a, z_a)$.

Now, according to Stone and Sipala [27], if the probability of being forced to a backtrack in a binary search tree (with consequent cutoff) is constant at any level, then an algorithm will visit the tree probabilistically in linear time with respect to the tree depth. Unfortunately this is a limiting case because any practical algorithm behaves in such a way that the cutoff probability increases, due to the accumulation of information, as the visit goes deeper. What can be done about this is to keep this increase as small as possible. We may apply these conclusions to our search tree since they are valid even if a nonbinary search tree is concerned. Therefore in view of the previous observation on the probabilities for a node to be nonfeasible in the PESP algorithm, the best thing to do is start with a minimal spanning tree $B$ and then consider sorted chords.

Tables 1, 2, and 3 report some experimental computations for the PESP algorithm of Fig. 3. The data of each instance have been randomly generated with the following rules: first the directed graph corresponding to the network PESP model has been generated with independent arc probabilities given by $d/(n-1)$, for all arcs $(i, j)$ with $i < j$, where $d$ is the desired average degree at each node. Some additional arcs have been possibly generated to make the network connected. Then the span constraints have been generated for all these arcs as $[\lfloor aT \rfloor, \lfloor (a+b)T \rfloor]$ where $a$ and $b$ are two independent random numbers uniformly generated between 0 and 1 and the period $T$ has been set to 100. The rounding is necessary to avoid round-off errors and consequent anomalous behaviour of some routines.

The values of $d$ have been set to: 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00. These values are the most significant ones, since for smaller values the algorithm runs almost without backtracking to a feasible solution, whereas for larger values the span constraints are so binding to make the depth of the search tree very small, and infeasibility is reported almost immediately. The values of $n$ have been set to: 25, 50, 75, 100, 150, 200. Twenty instances have been generated for each fixed value of $d$ and $n$.

Table 1 reports the average cpu time in milliseconds (on a VAX 11/780). The smaller figures appearing as subscripts and superscripts of the average times refer to the smallest and to the largest cpu time, respectively, over the twenty instances. A threshold value of 100 seconds cpu time was set to interrupt the execution of the algorithm. The number of instances out of twenty requiring more than 100 seconds and thus being interrupted is reported within brackets in the relative entry of the table if greater than zero. In case of interruption, the cpu time was set to 100 seconds and the instance considered infeasible.

Table 2 reports the average, the smallest, and the largest number of visited nodes of the search tree, and Table 3 reports the percentage of feasible instances over the same twenty instances.

The figures of Table 2 show a nondramatic increase in the number of visited nodes as the instance size increases, thereby confirming the previous theoretical considerations. The cpu times increase at a higher rate because on each node a shortest path problem has to be solved on a network whose size increases with the search tree level. However, average figures are poorly indicative of the behaviour of the algorithm: as is typical of NP-complete problems, the running times for homogeneous instances range over a wide spectrum of values. Whereas for most instances the running times are acceptable, once in a while an instance appears with a very long running time. Due to these reasons we also report the smallest and the largest running times to give a more precise idea of the behaviour of the algorithm.

It is fair to say that for instances with large spans, say almost as large as the period, the algorithm does not behave as satisfactory as for uniformly random spans. This is the case for instance of the Hamiltonian circuit problem solved through PESP (see Thm. 1). Moreover there are some scheduling problems where such large spans arise naturally as we shall see in §§ 2.7 and 3.4.

**2.5. The extended PESP (EPESP).** As we shall see in § 3, the presence of resources may complicate the periodic character of the scheduling problem by introducing different periods $mT$ (for integers $m$), which have to be taken into consideration simultaneously (see also for instance [6], [14], [21]). For instance, spans of the type $[d^-, d^+]_{mT}$ may be present in the model.

In principle, the PESP model can handle these situations as well since it is possible to consider the larger period $\hat{m}T$, with $\hat{m}$ the least common multiple of the integers $m$ and to transform each span $[d^-, d^+]_{mT}$ into

$$\bigcup_{k=1\cdots\hat{m}/m} [d^- + kmT,\ d^+ + kmT]_{\hat{m}T}$$

and to express in turn this union as an intersection of spans, as explained earlier.

However, this is unnecessarily expensive from a computational point of view because each of these span constraints produces several arcs, thus increasing the number of chords and expanding the search tree.

It is more convenient to refer directly to the network model of PESP and then to adapt with minor changes the previous algorithm to this case. More precisely we may define the Extended Periodic Event Scheduling Problem (EPESP) in the following way:

Given

— a period $T$;

— a network $(N, A)$;

— positive integers $m(a)$, $\forall a \in A$;

— real spans $D(a, z)$, $\forall a \in A$, $\forall z \in Z$ where $D(a, z) = [d^-(a) + zm(a)T, d^+(a) + zm(a)T]$ also denoted as $D(a, z) = [d^-(a), d^+(a)]_{m(a)T}$;

find a potential $u$ such that $v(a) \in D(a, z(a))$ for some $z(a) \in Z$, for all $a \in A$. Then $\tau(\varepsilon) = u(\varepsilon) \bmod T$.

Note the difference with PESP where $m(a) = 1$, for all $a$.

The practical utility of this model will be clear later when activities and resources will be taken into account. For the moment let us focus on a solution method for EPESP.

First let us note that the PESP algorithm is not adequate to solve EPESP since tree integrality does not necessarily hold for all spanning trees of the network $(N, A)$. In fact, if the instance is feasible, and $z(a) = z_1$ is a feasible integer value for the branch $a$ of a spanning tree $(N, B)$, and we replace $z_1$ with $z_2$, then the tension on an arc $c$ belonging to the cutset generated by $a$ changes by $(z_2 - z_1)m(a)T$. Since it is required that $d^-(c) \leqq v(c) - z(c)m(c)T \leqq d^+(c)$, the change in the tension $v(c)$ will not affect feasibility of the corresponding span constraint if $(z_2 - z_1)m(a)$ is an integer multiple of $m(c)$. So if we want to assign any integer value $z$ to the branch $a$ we must require that $m(c)$ divides $m(a)$.

Consider for instance the following example: $N = \{\varepsilon_1, \varepsilon_2\}, A = \{a_1, a_2\}, \varepsilon^-(a_1) = \varepsilon^-(a_2) = \varepsilon_1, \varepsilon^+(a_1) = \varepsilon^+(a_2) = \varepsilon_2, D(a_1) = [1, 5]_{10}, D(a_2) = [7, 8]_5$. Then if $B = \{a_2\}$ the algorithm will report infeasibility if $z(a_2)$ is fixed to an odd value and feasibility if $z(a_2)$ is fixed to an even value, whereas if $B = \{a_1\}$ the algorithm will report feasibility for any fixed value of $z(a_2)$.

In general terms, let us call *chord integrality* the condition for a given chord $c$ of a spanning tree that $m(c)$ divides all the integers $\{m(a)\}_{a \in C(c)}$, with $C(c)$ the circuit generated by $c$, and let us call *branch integrality* the condition for a given branch $a$ of a spanning tree that all $m(c)$ in the cutset generated by $a$ divide $m(a)$; then tree integrality holds for the given spanning tree if branch integrality is verified by all branches or, equivalently, chord integrality is verified by all chords. See Fig. 6.4 for an example of a spanning tree for which tree integrality holds.

Thus if a spanning tree satisfying tree integrality is available, the PESP algorithm can be used without modifications. If there is no spanning tree satisfying tree integrality (or if it is computationally hard to find one such tree), it is nevertheless possible to modify the PESP algorithm in the following way.

Let the root node of the search tree correspond to the problem $P(B', z_{B'})$ where $B' \subset B$ is a set of branches of a spanning tree $B$ for which branch integrality holds ($B'$ can be void). Then let the algorithm run as before with $\{c_1, \cdots, c_{q'}\} = B \backslash B'$ and $\{c_{q'+1}, \cdots, c_q\} = A \backslash B$. The only difference with the PESP algorithm consists in the generation of successor nodes at the first $q'$ levels. More precisely, when $c_i \in B \backslash B'$ the successor nodes of $P(J, z_J)$, with $J = B' \cup c_1 \cup \cdots \cup c_{i-1}$ correspond to the problems $P(J \cup c_i, z_{J \cup c_i})$ with $z_{c_i} = 1, \cdots, \hat{m}/m(c_i)$ and $\hat{m}$ the least common multiple of the $m(a)$ relative to the arcs $a$ belonging to the cutset generated by $c_i$ (including the arc $c_i$).

Note that $\hat{m}/m(c_i) = 1$ if branch integrality holds for $c_i$. So this modification is actually a generalization of the PESP algorithm. We have still to investigate whether it is more convenient from the computational point of view to find a set $B'$ of maximal cardinality, thus reducing the number of levels, or to find a set $B'$ with spans as narrow as possible, thus reducing the number of successors.

**2.6. Periodic events sequencing.** A set of periodic events cannot be ordered with respect to their schedules since these are elements of the group $R/T$. However a weaker concept of ordering is preserved if we consider how the occurrences of $m$ periodic events are displaced in a (real) time interval of length $T$. More formally, we say that $m$ real numbers $a_1, \cdots, a_m$ are *sequenced with respect to* $T$ if

$$(a_i + b) \bmod T \leqq (a_{i+1} + b) \bmod T$$

holds true for some real $b$ and for all $i = 1, \cdots, m - 1$ and we say that they are *strictly sequenced* if strict inequality holds in the above relationship.

Then we shall say that *m periodic events* $\varepsilon_1, \cdots, \varepsilon_m$ *are sequenced* or *are scheduled in sequence* (*strictly sequenced* or *scheduled in strict sequence*) if the real numbers $t(\varepsilon_1), \cdots, t(\varepsilon_m)$ are sequenced (strictly sequenced). Of course two scheduled periodic events are always sequenced and for $m$ scheduled periodic events exactly $m$ permutations of them correspond to strictly sequenced periodic events.

Constraints requiring certain subsets of periodic events to be sequenced in a definite way will appear naturally when we introduce resources in the model.

It is not difficult to impose the condition that $m$ periodic events have to be scheduled in sequence. The following lemma provides a key property.

LEMMA. *Let* $a_1, \cdots, a_m$ *be real numbers such that the numbers* $a_i \bmod T$ *are not all equal, and let* $z_{ij}$ *be integers so that* $0 \leqq a_j - a_i - z_{ij}T < T$. *Then the* TSP *with costs* $(-z_{ij})$ *has optimal value* 1 *and the optimal solution is exactly the permutation which sequences the* $a_i$*'s.*

*Proof.* Let $z_i$ be integers so that $0 \leqq a_i - z_iT < T$. Let $\alpha_i = a_i - z_iT$. Therefore we have

$$0 \leqq \alpha_j + z_jT - \alpha_i - z_iT - z_{ij}T < T \quad \text{i.e.,} \quad 0 \leqq \alpha_j - \alpha_i - (z_{ij} - z_j + z_i)T < T.$$

Note now that the TSP with costs $-(z_{ij} - z_j + z_i)$ is equivalent to the TSP with costs $(-z_{ij})$. However it is obvious that $-(z_{ij} - z_j + z_i)$ must be either 1, if $\alpha_j < \alpha_i$, or 0, if $\alpha_j \geqq \alpha_i$ and therefore, since the $\alpha_i$ are not all equal, the optimal solution is the one ordering the $\alpha_i$'s with optimal value 1. □

With the aid of the previous lemma, it is possible to handle a sequence or strict sequence condition by means of span constraints. Actually if the periodic events $\varepsilon_1, \cdots, \varepsilon_m$ have to be scheduled in sequence with respect to $T$ (besides other possible span constraints originally present in the problem), $m$ arcs

$$s_1 := (\varepsilon_1, \varepsilon_2), \cdots, s_m := (\varepsilon_m, \varepsilon_1)$$

forming a circuit $S$ have to be added to the network with spans

$$D(s_i, z) := [zT, T + zT].$$

If $\varepsilon_1, \cdots, \varepsilon_m$ have to be strictly sequenced, we have to add the following spans:

$$D(s_i, z) := [zT, T - U + zT]$$

where $U$ is the greatest commensurability unit of the instance data (if we suppose the data are rational numbers).

Then for any potential $u(\varepsilon_i)$ we compute $z_i$: $v(s_i) \in D(s_i, z_i)$ (the tie occurring if $u(\varepsilon_i) - u(\varepsilon_j) \bmod T = 0$ is broken by choosing the smaller $z$). From the lemma we know that the events $\varepsilon_1, \cdots, \varepsilon_m$ are scheduled in sequence if and only if $\sum_i z_i = -1$, unless $\Pi \cdot u(\varepsilon_i)$ are not all equal, in which case the events are trivially sequenced.

Therefore the algorithm runs as for a usual PESP with the only difference being that whenever $s_i \in S$ and $S \subset J \cup s_i$, i.e., the last arc of $S$ is appended to $J$, then the value $z(s_i)$, according to the previous results, has to be fixed to the value

$$z(s_i) = -1 - \sum_{j \neq i} z_{s_j}.$$

Apparently a sequence condition on $m$ periodic events introduces $m - 1$ new levels to the search tree (not counting the one with fixed $z$). Sometimes this added complexity can be reduced if the span $D(s_i)$ contains a previous span $D$ for the same pair of periodic events, in which case $D$ can freely subsume $D(s_i)$, as it happens for instance, in the problems dealt with in § 3.3.

For the tie to be broken for the choice of $z(s_i)$ mentioned above, it can be seen that the PESP algorithm itself will break the tie automatically as required and so there is no need to explicitly impose this extra condition.

**2.7. PESP and ΔTSP.** The results of the previous section and the reduction given by Theorem 1 show that there is a close connection between PESP and ΔTSP. In fact it is not difficult to see that for any instance of the symmetric ΔTSP with costs $c_{ij}$, the question "does there exist a tour with cost less than or equal to $T$?" is equivalent to the question about feasibility of the PESP instance with period $T$ and spans $[c_{ij}, T - c_{ij}]$ for all pairs of periodic events arbitrarily ordered. Incidentally this is an alternative proof of NP-completeness for PESP.

By this transformation a ΔTSP instance can be obviously solved via the PESP algorithm described in §§ 2.3 and 2.4. However, due to the particular spans involved in this case, that algorithm does not perform in a satisfactory way. We must point out that similar spans can occur for periodic scheduling problems (see § 3.4). Therefore, whenever this situation is faced, it seems to be advisable to solve the scheduling problem by using known techniques developed for ΔTSP.

THE PESP ALGORITHM.

```
compute minimal spanning tree $B$;
sort chords $c_1, \cdots, c_q$;
compute potentials on $B$;
                    {initialization search tree visit}
$J := B$; $i := 1$; status[1] := new; for $i := 1$ to $q$ do blocking_chord[$i$] := false; feasible := false; backtracking := false;
                    {search tree visit}
 {$z_i$ is the next integer to be guessed at level $i$. Each level can be in one of three states: new, all times when it is reached
  from above; the status new is immediately reset to alternating to guess integer values alternating away from the most
  central integer; when one of the two extreme infeasible nodes is visited the status is reset to one_way and integer
  values are looked for in one direction only.
  Shortest_path($\varepsilon_1, \varepsilon_2, J, \bar{v}$, feasible) is a routine computing the shortest path from $\varepsilon_1$ to $\varepsilon_2$ on the network $J$ via the Dijkstra's
  algorithm with an anticipated stop if a node is found whose distance from $\varepsilon_1$ exceeds $\bar{v}$; feasible is set false if and only
  if the distance from $\varepsilon_1$ to $\varepsilon_2$ is shorter than $\bar{v}$. }
while $1 \leqq i \leqq q$ do
begin
  repeat          {trying to find a feasible solution at level $i$}
   if status[$i$] = new
   then begin     {executed only if level $i$ is reached from level $i - 1$}
     status[$i$] := alternating;
     if $\exists \bar{z}$: $v(c_i) \in D(c_i, \bar{z})$
     then begin   {tension is already feasible}
       $z_i := \bar{z} + 1$;
       feasible := true;
       step[$i$] := 1;
       direction[$i$] := 1;
     end
     else begin   {tension is not feasible}
       compute $z^1 := \text{argmin}_z\{d^-(c_i) + zT - v(c_i) \geqq 0\}$;
       compute $z^2 := \text{argmin}_z\{v(c_i) - d^+(c_i) - zT \geqq 0\}$;
       if $d^-(c_i) + z^1T - v(c_i) \leqq v(c_i) - d^+(c_i) - z^2T$
       then begin {tension should be raised}
         $z_i := z^1$;
         direction[$i$] := 1;
       end
       else begin {tension should be lowered}
         $z_i := z^2$;
         direction[$i$] := $-1$;
       end;
       step[$i$] := 0;
     end;
   end
```

FIG. 3

```
        else begin        {either remaining in level i or reaching level i from below}
          if direction[i] = 1
          then shortest_path($\varepsilon^-(c_i)$,$\varepsilon^+(c_i)$,J,$d^-(c_i) + z_iT - v(c_i)$, feasible)
          else shortest_path($\varepsilon^+(c_i)$,$\varepsilon^-(c_i)$,J,$v(c_i) - d^+(c_i) - z_iT$, feasible);
          if feasible
          then begin
            update_potential;
            if status[i] = alternating
            then begin
              step[i] := step + 1;
              direction[i] := -direction[i];
            end;
            $z_i := z_i$ + direction[i]·step[i];
          end
          else begin
            for j := 1 to i - 1 do
              if $c_j \in$ blocking_circuit then blocking_chord[j] := true;
            case status[i] of
              alternating: begin
                                direction[i] := -direction[i];
                                $z_i := z_i$ + direction[i]·step[i];
                                step[i] := 1;
                                status[i] := one_way;
                              end;
              one_way:  backtracking:= true;
            end;
          end;
        end;
      until feasible or backtracking;
      if feasible          {going down one level}
      then begin
        J := J ∪ $c_i$;
        i := i + 1;
        status[i] := new;
        feasible := false;
      end;
      if backtracking     {accelerated backtracking}
      then begin
        repeat
          i := i - 1
          J := J\$c_i$;
        until (i = 1) or blocking_chord[i];
        if blocking_chord[i]
        then for j := 1 to i do blocking_chord [j] := false
        else i := i - 1;
        backtracking := false;
      end;
    end;
    if i = 0 then output ("PESP is infeasible");
    if i := q + 1 then output solution;
```

FIG. 3 (*Continued*)

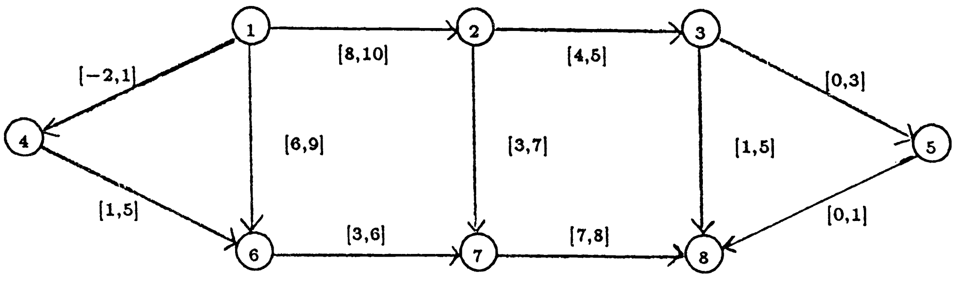


FIG. 4.1. *A* PESP *instance*.

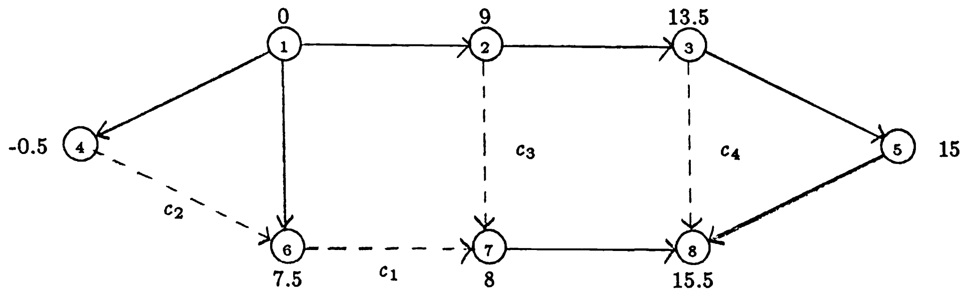


FIG. 4.2. *Spanning tree and initial potentials.*

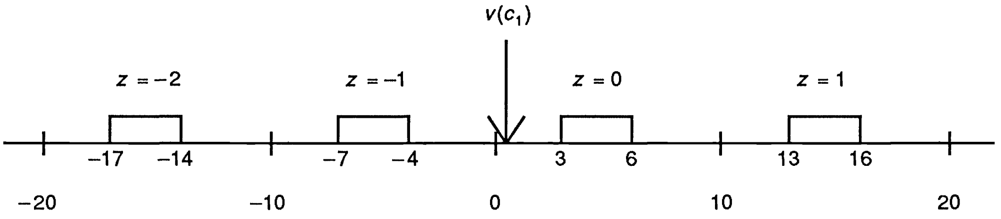


FIG. 4.3. *Admissible $z(c_1)$ values.*

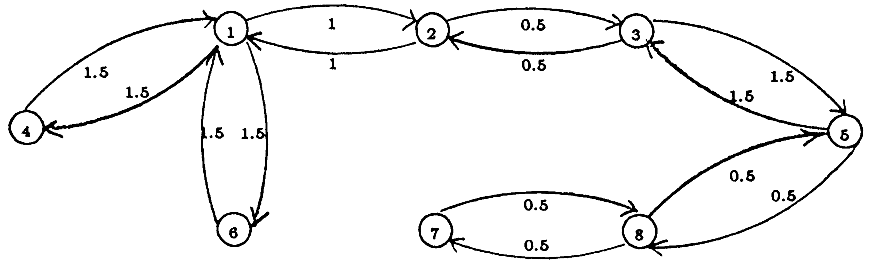


FIG. 4.4. *Graph $(N, B')$.*

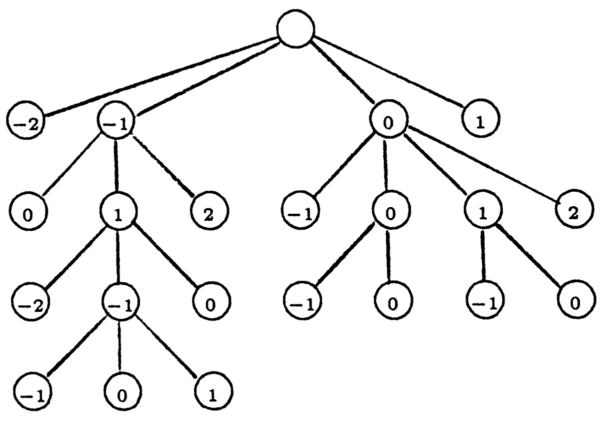


FIG. 4.5. *Full search tree.*

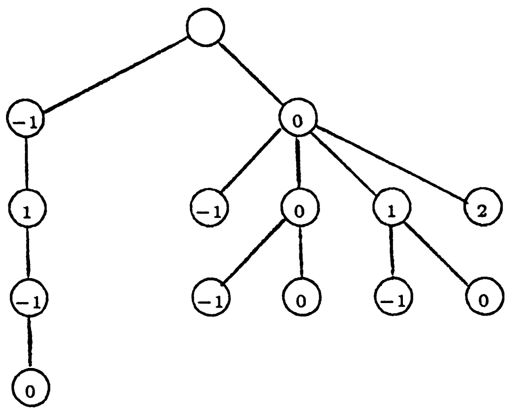
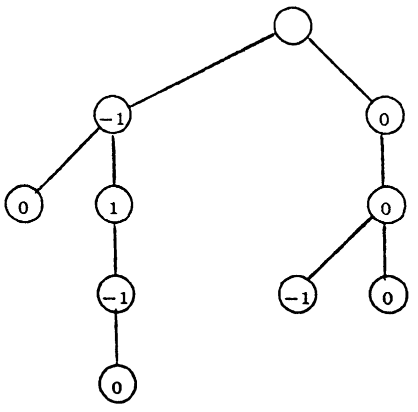


FIG. 4.6. *Nodes visited by naive algorithm.*



Solution : $\tau(\epsilon) = (0\quad 8\quad 2\quad 8\quad 3\quad 9\quad 5\quad 3)$

FIG. 4.7. *Nodes visited with accelerated backtracking.*

TABLE 1
Cpu *times in milliseconds.*

| | 25 | 50 | 75 | 100 | 150 | 200 |
|---|---|---|---|---|---|---|
| 1.50 | $59^{120}_{40}$ | $153^{210}_{120}$ | $303^{390}_{240}$ | $522^{940}_{410}$ | $1066^{1170}_{950}$ | $1817^{2150}_{1570}$ |
| 2.00 | $76^{270}_{50}$ | $184^{220}_{150}$ | $384^{590}_{310}$ | $671^{1080}_{520}$ | $1923^{12290}_{1030}$ | $2274^{5120}_{1670}$ |
| 2.50 | $98^{240}_{60}$ | $265^{830}_{160}$ | $432^{550}_{340}$ | $1098^{7800}_{560}$ | $3522^{31970}_{1140}$ | $2995^{22690}_{1530}$ |
| 3.00 | $124^{250}_{50}$ | $334^{1010}_{190}$ | $1534^{10830}_{370}$ | $2247^{16330}_{560}$ | $5680^{40240}_{1220}$ | $17273^{100000}_{2370}$ (2) |
| 3.50 | $156^{470}_{60}$ | $538^{3290}_{190}$ | $1792^{17880}_{390}$ | $5953^{49400}_{770}$ | $8557^{42850}_{1440}$ | $45293^{100000}_{2600}$ (8) |
| 4.00 | $169^{440}_{70}$ | $740^{6270}_{240}$ | $1577^{7620}_{470}$ | $7930^{99000}_{770}$ | $17748^{100000}_{1820}$ (1) | $34864^{100000}_{2880}$ (5) |
| 4.50 | $171^{520}_{9}$ | $969^{7030}_{270}$ | $1172^{5550}_{510}$ | $1986^{19840}_{830}$ | $26393^{100000}_{1800}$ (4) | $5578^{20230}_{3320}$ |
| 5.00 | $186^{770}_{100}$ | $705^{2820}_{310}$ | $2451^{7550}_{620}$ | $2755^{15950}_{960}$ | $15177^{100000}_{2170}$ (2) | $10678^{77740}_{3940}$ |

TABLE 2
*Number of visited nodes.*

| | 25 | 50 | 75 | 100 | 150 | 200 |
|---|---|---|---|---|---|---|
| 1.50 | $8_{3}^{16}$ | $13_{3}^{28}$ | $18_{3}^{28}$ | $26_{10}^{41}$ | $42_{32}^{56}$ | $55_{37}^{74}$ |
| 2.00 | $12_{3}^{25}$ | $19_{3}^{29}$ | $32_{6}^{45}$ | $40_{3}^{61}$ | $59_{3}^{166}$ | $71_{3}^{113}$ |
| 2.50 | $16_{3}^{45}$ | $27_{3}^{60}$ | $38_{3}^{56}$ | $45_{3}^{220}$ | $86_{3}^{651}$ | $86_{3}^{219}$ |
| 3.00 | $16_{3}^{45}$ | $33_{3}^{68}$ | $68_{3}^{449}$ | $73_{3}^{558}$ | $89_{3}^{553}$ | $213_{3}^{1132}$ (2) |
| 3.50 | $23_{3}^{86}$ | $37_{3}^{269}$ | $51_{3}^{449}$ | $63_{3}^{315}$ | $114_{3}^{535}$ | $306_{3}^{818}$ (8) |
| 4.00 | $20_{3}^{89}$ | $35_{3}^{350}$ | $34_{3}^{189}$ | $159_{3}^{2039}$ | $142_{3}^{1050}$ (1) | $194_{3}^{746}$ (5) |
| 4.50 | $14_{3}^{47}$ | $45_{3}^{292}$ | $18_{3}^{114}$ | $17_{3}^{229}$ | $243_{3}^{1005}$ (4) | $15_{3}^{100}$ |
| 5.00 | $14_{3}^{78}$ | $27_{3}^{140}$ | $45_{3}^{170}$ | $27_{3}^{210}$ | $100_{3}^{857}$ (2) | $32_{3}^{308}$ |

TABLE 3
*Percentage of feasible problems.*

| | 25 | 50 | 75 | 100 | 150 | 200 |
|---|---|---|---|---|---|---|
| 1.50 | 95 | 90 | 85 | 95 | 100 | 100 |
| 2.00 | 80 | 80 | 95 | 90 | 70 | 75 |
| 2.50 | 70 | 60 | 80 | 50 | 55 | 65 |
| 3.00 | 50 | 75 | 55 | 45 | 35 | 35 (2) |
| 3.50 | 50 | 40 | 20 | 30 | 35 | 10 (8) |
| 4.00 | 25 | 20 | 5 | 10 | 0 (1) | 0 (5) |
| 4.50 | 15 | 20 | 0 | 0 | 5 (4) | 0 |
| 5.00 | 5 | 10 | 0 | 0 | 0 (2) | 0 |

## 3. Activities and resources.

**3.1. Basic definitions.** A *periodic activity* $\gamma$ is an ordered pair of periodic events $(\varepsilon^-(\gamma), \varepsilon^+(\gamma))$, corresponding to its beginning and ending. A periodic activity $\gamma$ is said to be scheduled if both $\varepsilon^-(\gamma)$ and $\varepsilon^+(\gamma)$ are scheduled and a nonnegative real number $x(\gamma)$, i.e., its *duration*, is assigned in such a way that

$$x(\gamma)=t(\varepsilon^+(\gamma))-t(\varepsilon^-(\gamma))+z(\gamma)T \geqq 0 \text{ for some } z(\gamma)\in Z.$$

We shall be concerned with a finite set $\Gamma$ of periodic activities.

The $p$th occurrence of the periodic activity $\gamma$ is the activity starting at $t(e^-(\gamma, p))$ and ending at $t(e^-(\gamma, p)) + x(\gamma)$. It is denoted by $(\gamma, p)$.

It is interesting to deal with activities using resources. By *resources* we mean a finite set $\mathcal{R}$, where each resource $R \in \mathcal{R}$ is in turn a finite set of *resource units* of some definite type. To each periodic activity $\gamma$, a set $\mathcal{R}(\gamma) \subset \mathcal{R}$ is associated, which corresponds to the resources needed by the activity $\gamma$. Correspondingly, we define $\Gamma(R) = \{\gamma\colon R \in \mathcal{R}(\gamma)\}$, that is, the set of activities needing the resource $R$.

The resource units constitute a set of indistinguishable units and so any activity needing a particular resource may use any unit of it. Therefore we have to solve at the same time the problem of scheduling the activities and the problem of assigning the activities to the resource units. Formally we define as a *resource assignment* for the resource $R$ and denote it by $RA_R$ *an assignment of* $(\gamma, p)$ *to exactly one* $r \in R$ *for all* $R \in \mathcal{R}(\gamma)$ *and all* $p \in Z$. For ease of presentation we shall occasionally simplify the notation by dropping the dependence on the particular resource, if it is not necessary to emphasize this dependence.

A resource assignment $RA$ can be viewed as a two-dimensional array, with rows corresponding to periodic activities and columns corresponding to occurrences, whose $(\gamma, p)$ entry refers to some resource unit $r = RA(\gamma, p) \in R$. By $RA(p)$ we denote the $p$th column of $RA$ (see Fig. 5).

For any two periodic activities $\gamma_i$ and $\gamma_j$ using the same resource, let $s(\gamma_i, \gamma_j)$ be the least amount of time needed by a resource unit released by activity $\gamma_i$ to be available for the activity $\gamma_j$. This time will be called the *set-up time*.

Depending on how the activities are scheduled, the resource assignment may or may not be feasible, where infeasibility is intended as the simultaneous request in the real time of the same resource unit by two different activities, or also by two different occurrences of the same activity (if for instance the duration of the activity is longer than the period).

DEFINITION. An $RA$ is *feasible* if $RA(\gamma_i, p) = RA(\gamma_j, q)$ and $(\gamma_i, p) \neq (\gamma_j, q)$ imply that either $t(e^-(\gamma_i, p)) + x(\gamma_i) + s(\gamma_i, \gamma_j) \leqq t(e^-(\gamma_j, q))$ or $t(e^-(\gamma_j, q)) + x(\gamma_j) + s(\gamma_j, \gamma_i) \leqq t(e^-(\gamma_i, p))$.

We shall model the problem of finding a feasible schedule (feasible with respect to the span constraints) together with a feasible resource assignment (as defined above) through a feasibility problem with respect to span constraints only, thereby exploiting the results of § 2. In order to achieve this result, we first need to show that the resources are used in a peculiar periodic way. Thereafter we shall show that feasibility of a resource assignment can be expressed via span constraints.

First note that, due to the finiteness of $R$, there must be two occurrences $p$ and $p'$ such that $RA(p) = RA(p')$. We shall make the following assumption.

*Assumption.* $RA(p + z) = RA(p' + z)$ for all $z \in Z$.

This way the resource assignment exhibits a periodic behaviour with period $T' = |p' - p| T = mT$. We call this time period the *resource cycle*. Besides this cyclic aspect, a resource assignment is also inherently periodic with respect to $T$ as shown in the following theorem.

THEOREM 3. *For any fixed activity schedule there exists a resource assignment $RA$, minimizing the number of used resource units, such that $RA(p + 1)$ is obtained from $RA(p)$ by applying a fixed permutation $PR$ (the same for all $p$'s) to the elements of $R$.*

Hence a resource assignment is determined by the following two unidimensional arrays of lengths $|\Gamma|$ and $|R|$, respectively:

$T = 10$

$\mathscr{R} = \{R\}$, $\quad R = \{r_1, r_2, r_3, r_4, r_5\}$, $\quad \Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6\}$

$\Gamma(R) = \Gamma$, $\quad \mathscr{R}(\gamma_i) = \mathscr{R} \quad \forall i$, $\quad s(\gamma_i, \gamma_j) = 0$

Activity schedule

| | $t(\varepsilon^-)$ | $t(\varepsilon^+)$ | $x(\gamma)$ | $z(\gamma)$ |
|---|---|---|---|---|
| $\gamma_1$ | 0 | 5 | 5 | 0 |
| $\gamma_2$ | 7 | 2 | 5 | 1 |
| $\gamma_3$ | 2 | 3 | 1 | 0 |
| $\gamma_4$ | 4 | 8 | 4 | 0 |
| $\gamma_5$ | 3 | 6 | 13 | 1 |
| $\gamma_6$ | 7 | 1 | 4 | 1 |

$RA(0) = [r_1, r_1, r_2, r_2, r_4, r_5]$

Resource assignment

| $p$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $\gamma_1$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ |
| $\gamma_2$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ |
| $\gamma_3$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ |
| $\gamma_4$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ | $r_3$ | $r_1$ | $r_2$ |
| $\gamma_5$ | $r_4$ | $r_5$ | $r_4$ | $r_5$ | $r_4$ | $r_5$ | $r_4$ |
| $\gamma_6$ | $r_5$ | $r_4$ | $r_5$ | $r_4$ | $r_5$ | $r_4$ | $r_5$ |

$PR = (2, 3, 1, 5, 4) = (1, 2, 3)(4, 5)$

Resource plan

$Q = \{\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\} \{\gamma_5, \gamma_6\}\}$ $\quad M = (3, 2)$ $\quad PA = (2, 3, 4, 1, 6, 5) = (1, 2, 3, 4)(5, 6)$

FIG. 5. *Example of resource assignment and resource plan.*

1) a particular column $RA(p)$ of the resource assignment;

2) a permutation $PR$ on $R$.

*Proof.* We limit ourselves to outlining the proof; the reader will have no difficulties in filling in the necessary details. First we need to define a *periodic bipartite graph*. We say that $(V, U, E)$ is a periodic bipartite graph if $|V| = |U| = n = mk$ with $m$ and $k$ positive integers such that

$$(v_i, u_j) \in E \Leftrightarrow (v_{(i+k)\bmod n}, u_{(j+k)\bmod n}) \in E.$$

A *weighted* periodic bipartite graph has weights obeying a similar relationship.

We build a complete weighted periodic bipartite graph to represent feasible resource assignments with resource cycle $mT$ and $k = |\Gamma(R)|$. In particular $v_{jk+i} \in V$ corresponds to the event related to the ending of the $j$th occurrence (within the resource cycle) of the $i$th activity of $\Gamma(R)$. Similarly, vertices of $U$ are related to beginnings of activity occurrences. Two occurrences of the same activity distant $m$ periods are identified, according to the previous assumption.

Recall that all activities have been scheduled. Assigning a resource unit, whose use by the $j$th occurrence of the $i$th activity has just terminated, to the $p$th occurrence of the $q$th activity, is equivalent to matching the vertices $v_{jk+i}$ and $u_{pk+q}$ of $G$. The weight to be attached to this arc is simply given by the actual idle time the resource unit would spend during this transfer. The weights have an obvious periodic character.

The total weight of any matching plus the activity durations is equal to a certain multiple of the resource cycle. This multiple is also the number of resource units necessary to produce a feasible resource assignment according to the given matching.

The given resource assignment (with resource cycle $mT$) corresponds to a definite matching on the periodic graph just defined. Due to the previous observation, finding a matching of minimum weight is equivalent to finding a possibly different resource assignment with less or equal resource units. We are therefore interested in finding a matching of minimum weight in a weighted complete periodic bipartite graph.

Our claim is that there exists an optimal matching which is a periodic subgraph of $(V, U, E)$ (with the same $m$ and $k$). We recall that a weighted bipartite matching can be solved through a primal-dual method by solving a certain number of cardinality bipartite matching problems (see [20]).

Our subclaim is that there exists a maximum cardinality matching for a periodic bipartite graph which is periodic as well. Let us suppose that a periodic matching is given; then if it is not maximum there exist augmenting paths. Since the matching is periodic these paths are periodic as well. The idea is to augment simultaneously all these paths in order to have another periodic matching of larger cardinality. However this is possible only if the paths do not intersect. So we want to prove that there always exists a periodic set of nonintersecting augmenting paths. Therefore let us suppose that the augmenting path $v_i \rightarrow u_j$ intersects the augmenting path $v_{i+k} \rightarrow u_{j+k}$ with $v_{i+k}$ nearer to the intersection than $v_i$. Then we may "exchange" the paths in the intersection to form a new augmenting path $v_{i+k} \rightarrow u_j$. By repetitively doing so for all augmenting paths, we are left with a new periodic set of augmenting paths plus an alternating circuit intersecting all augmenting paths. This circuit can be dropped and so we have a set of periodic augmenting paths with fewer (if any) intersections than before. Now it is easy to prove the subclaim.

That the claim is true is also easily established by recalling the details of the Hungarian algorithm [20].

The optimal matching produces a resource assignment whose resource cycle is, in general, a positive multiple of $m$. Obviously it does not require more resource units over this larger resource cycle than the resource assignment we started with.

The fact that the optimal matching is periodic suffices to prove the theorem. □

In order to grasp the feeling of the theorem, suppose that a specified resource unit is used by a set $\Gamma' \subset \Gamma$ of activities, that is the resource unit is released by some activity in $\Gamma'$ and then is resumed by some other activity of $\Gamma'$ and so on, always in a cyclic way, in view of the previous assumption. The time necessary for the resource unit to be used by all activities in $\Gamma'$, and to be again available for the first one, need not be equal to one period; in fact it can be any value $mT$ with $m$ positive integer. The situation can be viewed as if the resource unit were travelling through the activities of $\Gamma'$ completing the tour in $m$ periods. If $m > 1$, since each periodic activity must be restarted after one period, there must be another resource unit ready to be used by the first activity in $\Gamma'$ at the beginning of the second period. In principle there is no reason why this second resource unit travels through the activities in the same sequence as the first unit did one period earlier. What the theorem says is that it is optimal, in terms of number of used resource units, that this second unit (and also all possible other ones) is used by the activities in $\Gamma'$ in the same sequence as was the first unit.

Therefore $m$ resource units must be circulating in the tour equally spaced of one period. This fact establishes that the number of periods necessary for a set $\Gamma'$ of activities to be completed by one resource unit is equal to the number of resource units needed by $\Gamma'$.

Of course there may be several disjoint subsets $\Gamma_j$ of activities, each one of them using $m_j$ resource units in a cyclic way. Hence the permutation transforming $RA(p)$ into $RA(p+1)$ is the direct product of some cyclic permutations of cardinality $m_j$. The resource cycle is then the least common multiple of the $m_j$'s times $T$. The presence of resources therefore introduces several periods which have to be taken into account simultaneously, and this is the very point where the EPESP model is called for.

Due to the periodic behaviour of the resource units, a resource assignment can be also expressed in terms of resources rather than in terms of activities. In fact, according to the previous results, for each resource $R$ there exists a permutation $PA(R)$ on $\Gamma(R)$ specifying the sequences of activities using the same resource units. The permutation $PA(R)$, being the direct product of some cyclic permutations $PA_1(R), \cdots, PA_{q(R)}(R)$, defines a partition $Q(R) = \{\Gamma_1(R), \cdots, \Gamma_{q(R)}(R)\}$ of $\Gamma(R)$ and to each $Q(R)$ an array $M(R) = \{m_1(R), \cdots, m_{q(R)}(R)\}$ of positive integers is associated such that $\sum_i m_i(R) = |R|$ (again see Fig. 5).

Hence it is possible to express an assignment of resources to activities by specifying $Q(R)$, $M(R)$, and $PA_j(R)$ for each resource $R$. Indeed this is the way we shall relate resources and activities and this will allow expression of the relationship among resources and periodic events by span constraints of the type described in § 2. With this respect we shall use the following terminology:

A *complete resource plan* is a specification concerning $Q(R)$, $M(R)$, and $PA_j(R)$.

A *partial resource plan* is a specification concerning only $Q(R)$ and $M(R)$.

Of course any resource plan is completely determined by a resource assignment together with an activity schedule, and conversely, any resource assignment is completely determined by a resource plan together with an activity schedule. However, if the activity schedule is not specified neither a resource assignment determines a resource plan nor vice versa.

**3.2. Resource planning with a fixed activity schedule.** The results of this section can be considered a corollary of Theorem 3. In fact the periodic behaviour of the resource units, implicitly assumed throughout this section, was established by that theorem.

In this section we consider the problem of finding a feasible $RA$ minimizing the number of resource units needed for a fixed activity schedule. As was seen in the previous section, the number of needed resource units is equal to $\sum_j m_j(R)$ for each resource $R$.

This problem can be solved by finding permutations $PA(R)$ of minimum cost, with the cost directly related to the $m_j(R)$'s. Of course the problem directly splits into separated problems, one for every resource.

For each resource $R$ we build a graph $(V, E)$ where $V = V^- \cup V^+$ with $V^- = \{\varepsilon^-(\gamma) : \gamma \in \Gamma(R)\}$, $V^+ = \{\varepsilon^+(\gamma) : \gamma \in \Gamma(R)\}$, and $E = A \cup B$ with $A$ the set of arcs from $V^-$ to $V^+$ corresponding to the activities in $\Gamma(R)$, and with $B$ the complete set of arcs from $V^+$ to $V^-$ corresponding to assignments of one resource unit from one activity to another one. That is, each arc $(\varepsilon^+(\gamma_i), \varepsilon^-(\gamma_j)) \in B$ expresses the fact that resource units may be assigned in sequence to the activities $\gamma_i$ and $\gamma_j$.

Then let us define

$$x(\gamma_i, \gamma_j) = \min_{x \in Z} \{t(\varepsilon^-(\gamma_j)) - t(\varepsilon^+(\gamma_i)) + zT \geqq s(\gamma_i, \gamma_j)\}$$

and let $z(\gamma_i, \gamma_j)$ be the argument yielding the minimum.

To arcs in $A$ the costs $x(\gamma)$ are assigned whereas the costs $x(\gamma_i, \gamma_j)$ are assigned to the arcs in $B$. The problem is then to find a set $C = \bigcup C_j$ of arc disjoint directed circuits in $(V, E)$ of minimum cost and such that $A \subset C$. In fact

$$\left(\sum_j m_j\right)T = \sum_A x(\gamma) + \sum_{B \cap C} x(\gamma_i, \gamma_j) = \left(\sum_A z(\gamma) + \sum_{B \cap C} z(\gamma_i, \gamma_j)\right)T.$$

It is immediate to see that this is equivalent to a weighted bipartite matching problem for the complete bipartite graph $(V^+, V^-, B)$ with costs given by the integers $z(\gamma_i, \gamma_j)$. From the optimal matching $\hat{B}$, the permutation $PA(R)$ is immediately derived and obviously

$$\sum_j m_j = \sum_A z(\gamma) + \sum_{\hat{B}} z(\gamma_i, \gamma_j),$$

with

$$m_j = \sum_{A \cap C_j} z(\gamma) + \sum_{\hat{B} \cap C_j} z(\gamma_i, \gamma_j).$$

So the solution of the problem consists of several resource cycles $C_j$ (possibly just one circuit) each associated to a certain multiple $m_j$ of the basic period $T$. It may be interesting to note that if we want a solution consisting of one resource cycle, the problem is no longer a weighted assignment but it becomes a TSP. On the other hand if we want a solution in which each resource cycle is long $T$, a circular graph coloring problem is produced which is also NP-complete [11]. Note also that if $x(\gamma_i) < T$, for all $i$ and $s(\gamma_i, \gamma_j) = 0$, for all $i, j$, then $z(\gamma_i, \gamma_j) \in \{0, 1\}$ and so the weighted bipartite matching problem actually becomes a simpler cardinality problem.

**3.3. Activity scheduling with a fixed complete resource plan.** The problem considered in this section is the one of scheduling a given set of periodic events subject to some span constraints of the type described in § 2.1, with the additional condition that some (or even all) events are starting and/or ending events of activities for which a complete resource plan has been fixed and is required to be feasible. To say that a resource plan has been fixed means that for each resource $R$ the partition $Q(R)$, the permutation $PA(R)$ and the arrays $M(R)$ are fixed to some definite values.

In order to have a feasible resource assignment, we have to add the condition that the events referring to the activities in $\Gamma_j(R)$ have to be scheduled in sequence with respect to $m_j(R)T$ (see §§ 2.5 and 2.6), according to the sequence given by $PA_j(R)$. So it is just a matter of applying the results of § 2.6. To be more specific, let $\Gamma_j(R) =$

$\{\gamma_1, \cdots, \gamma_p\}$ with activities labelled according to the sequence. For the sake of simplicity let us assume that the activity durations $x(\gamma_i)$ are fixed, so one periodic event (say the starting event) $\varepsilon_i$ is sufficient to identify the activity $\gamma_i$. Then the periodic events $\varepsilon_1, \cdots, \varepsilon_p$ have to be scheduled in sequence with respect to $m_j(R)T$.

Furthermore, feasibility of the resource assignment is obtained by imposing, for each pair of successive events,

$$a_i := (\varepsilon_i, \varepsilon_{(i+1)\mathrm{mod}p}) \qquad i = 1, \cdots, p,$$

and also imposing the span constraint given by

$$v(a_i) \in [x(\gamma_i) + s(\gamma_i, \gamma_{(i+1)\mathrm{mod}p}) + zm_j(R)T,\ W + x(\gamma_i) + s(\gamma_i, \gamma_{(i+1)\mathrm{mod}p}) + zm_j(R)T]$$

for some $z \in Z$, where the quantities $s(\gamma_i, \gamma_j)$ have been defined in § 3.1 and $W$ is computed as

$$W := m_j(R)T - \sum_{i=1}^{p} x(\gamma_i) + s(\gamma_i, \gamma_{(i+1)\mathrm{mod}p}).$$

The above span constraints clearly subsume the ones relative to the sequencing condition.

This is an EPESP with the added feature of sequencing (an example is shown in Fig. 6). Therefore a possible algorithm to solve it is the one described in § 2.5 with the modification pointed out in § 2.6. There is no conceptual difficulty in dealing with variable activity durations; it is just a matter of considering the ending events as well and of sequencing the events in the natural way with possible span constraints between the starting and the ending events.

Some particular cases are worth mentioning. If $|\mathcal{R}(\gamma)| \leqq 1$, that is, if each periodic activity requires at most one resource, then the sequence circuits are disjoint and therefore, since the span constraints involving multiples of the period $T$ refer to arcs in the sequence circuits, the tree integrality condition expressed in §§ 2.3 and 2.5 is automatically satisfied for any spanning tree containing all arcs but one of every sequence circuit.

If $|\Gamma(R)| = 2$, that is only two periodic events, corresponding to the starting events of two activities $\gamma_i$, $\gamma_j$, have to be sequenced, the following span for $(\varepsilon^-(\gamma_i), \varepsilon^-(\gamma_j))$ suffices to model a feasible resource assignment

$$[x(\gamma_i) + s(\gamma_i, \gamma_j), mT - x(\gamma_j) - s(\gamma_j, \gamma_i)]_{mT}.$$

In Fig. 6 an example is given with seven periodic activities of fixed duration and no set-up times. Three resources are involved consisting of three, two, and three units respectively (labeled $r_1$–$r_8$). The first four activities need the first and third resource (i.e., any one unit of both the first and the third resource), and the remaining activities need the second and the third resource. These are the problem data for which a feasible activity schedule together with a feasible resource assignment is looked for.

We suppose that the resource plan has been fixed in the following way: the resource cycle of resource $R_1$ consists of one cycle, namely $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4 \rightarrow \gamma_1$, lasting three periods since $|R_1| = 3$. Correspondingly, four arcs are given on the network with span $[x(\gamma_i), W + x(\gamma_i)]_{30}$ for the arc outgoing from node $i$ with $W = 15$, i.e., the period minus the sum of the durations along the cycle. Similarly the resource cycle of resource $R_2$ consists of the single cycle $\gamma_5 \rightarrow \gamma_6 \rightarrow \gamma_7 \rightarrow \gamma_5$ lasting two periods and, correspondingly, three arcs are given on the network with appropriate spans.

The resource $R_3$ is used in a different way: its three units go over three independent cycles, necessarily lasting one period. In particular one of the cycles (there is no need of explicitly assigning cycles to resource units since these are indistinguishable) involves

$\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7\}$
$x(\gamma_1) = 4,\quad x(\gamma_2) = 3,\quad x(\gamma_3) = 5,\quad x(\gamma_4) = 3,\quad x(\gamma_5) = 4,\quad x(\gamma_6) = 5,\quad x(\gamma_7) = 4$
$s(\gamma_i, \gamma_j) = 0$
$\mathcal{R} = \{R_1, R_2, R_3\}$
$R_1 = \{r_1, r_2, r_3\},\quad R_2 = \{r_4, r_5\},\quad R_3 = \{r_6, r_7, r_8\}$
$\mathcal{R}(\gamma_1) = \mathcal{R}(\gamma_2) = \mathcal{R}(\gamma_3) = \mathcal{R}(\gamma_4) = \{R_1, R_3\}$
$\mathcal{R}(\gamma_5) = \mathcal{R}(\gamma_6) = \mathcal{R}(\gamma_7) = \{R_2, R_3\}$
$\Gamma(R_1) = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\},\quad \Gamma(R_2) = \{\gamma_5, \gamma_6, \gamma_7\},\quad \Gamma(R_3) = \Gamma$

FIG. 6.1. *Activity data.*

$Q(R_1) = \{\Gamma(R_1)\},\quad Q(R_2) = \{\Gamma(R_2)\},\quad Q(R_3) = \{\{\gamma_1, \gamma_6\}, \{\gamma_2, \gamma_7, \gamma_4\}, \{\gamma_5, \gamma_3\}\}$
$PA(R_1) = (1, 2, 3, 4),\quad PA(R_2) = (5, 6, 7),\quad PA(R_3) = (1, 6)(2, 4, 7)(5, 3)$ (cyclic notation)
$m(R_1) = 3,\quad m(R_2) = 2,\quad m_1(R_3) = m_2(R_3) = m_3(R_3) = 1$

FIG. 6.2. *Complete resource plan.*

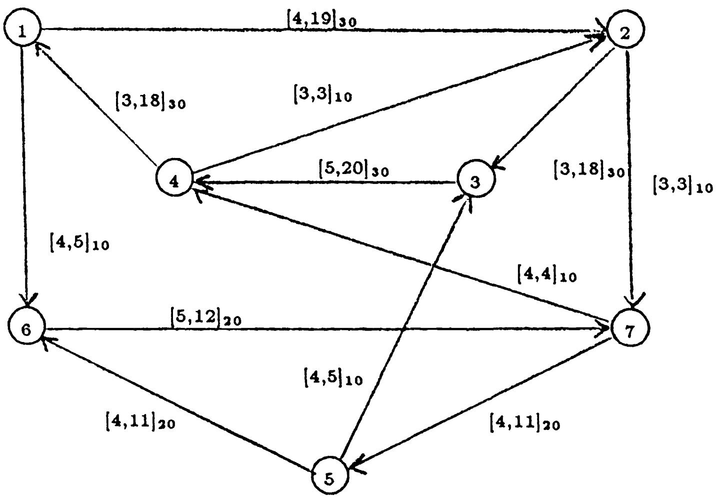


FIG. 6.3. EPESP *network for activity scheduling with fixed resource plan.*

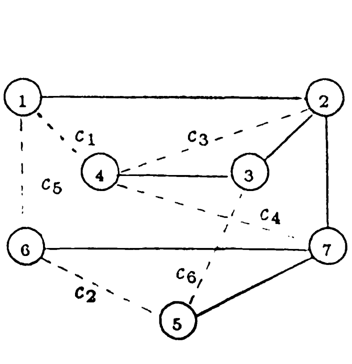


FIG. 6.4. *Spanning tree satisfying tree integrality.*

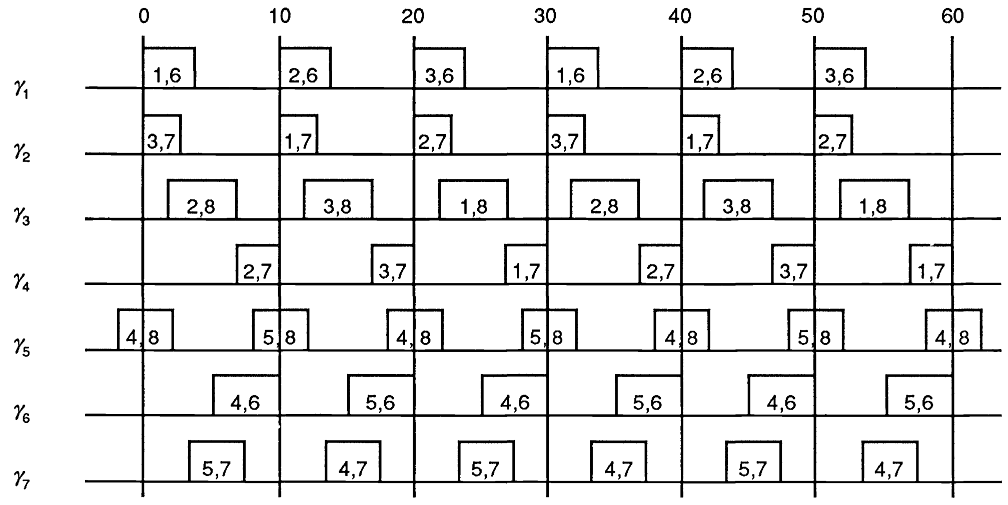


FIG. 6.5. *Solution.*

the activities $\gamma_2 \rightarrow \gamma_7 \rightarrow \gamma_4 \rightarrow \gamma_2$ (in this order) raising the corresponding arcs and spans on the network. The other two cycles involve only two activities each, namely $(\gamma_1, \gamma_6)$ and $(\gamma_3, \gamma_5)$. Since two activities are always sequenced only one arc between the activities suffices to model a feasible resource assignment, as was already pointed out.

The resulting network is shown in Fig. 6.3. A spanning tree satisfying tree integrality is shown in Fig. 6.4. In problems involving sequence conditions it is not convenient to sort the chords for increasing span lengths; in fact, since some span constraints have a fixed value for $z$ due to the sequence condition $\sum z_i = -1$, (and hence they correspond to a level with one successor only in the search tree) it is more convenient to have these span constraints at the highest possible levels. So in the example $c_1$ and $c_2$ are the chords closing the cycles of resources $R_1$ and $R_2$, respectively. In order to fulfill the sequence condition, since all arcs of the cycle but one belong to the spanning tree in both cases,

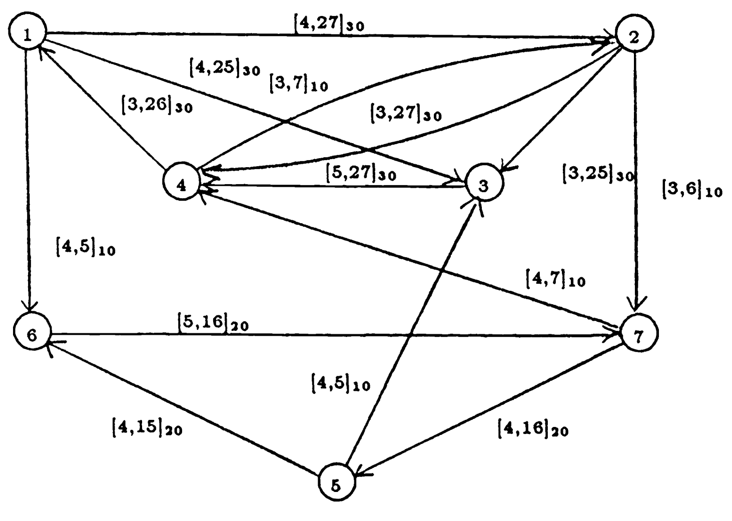


FIG. 6.6. EPESP *network for activity scheduling with fixed partial resource plan.*

it suffices to set $D(c_1) = [-27, -12]$ and $D(c_2) = [-16, -9]$ which correspond to the condition $\sum z_i = -1$.

As far as the resource cycle $\gamma_2 \rightarrow \gamma_7 \rightarrow \gamma_4 \rightarrow \gamma_2$ is concerned, only one arc of the cycle belongs to the spanning tree with value $z = 0$. So it is required $z(c_3) + z(c_4) = -1$. It can be seen that when the search tree level corresponding to $c_3$ is reached a first guess $z(c_3) = -2$ is made. This implies that in the next level $z(c_4) = 1$. It turns out that this value yields a feasible potential and so the visit of the search tree may proceed to deeper levels. For this particular example the final solution is obtained without backtracking. It is displayed in the form of a GANTT diagram in Fig. 6.5; the numbers inside the boxes refer to the resource units.

**3.4. Activity scheduling with a fixed partial resource plan.** A drawback of the model described in the previous section is that a very detailed description of the resource plan is required to be given in advance in order to schedule all events. In practical applications any resource plan is all right as long as it is feasible and consistent with the available number of resource units. Therefore we describe in this section a model which is relaxed with respect to the input data requirement but is computationally more expensive.

With respect to the problem of the previous section, the way the activities in each $\Gamma_j(R)$ have to be sequenced is no longer part of the problem data but a variable to be determined. Let us suppose again for the sake of simplicity that the activity durations are fixed so that the starting events $\varepsilon_1, \cdots, \varepsilon_p$ identify the activities in $\Gamma_j(R)$. Then the following span constraints between *all* (unordered) pairs $\{\varepsilon_h, \varepsilon_k\}$, $h = 1, \cdots, p-1$, $k = h+1, \cdots, p$

$$[x(\gamma_h)+s(\gamma_h, \gamma_k), mT-x(\gamma_k)-s(\gamma_k, \gamma_h)]_{mT}$$

correspond to the requirement of a feasible resource assignment.

In terms of the network model of the EPESP, these span constraints form a certain number of cliques. Scheduling the activities on each clique is a problem comparable to a TSP as seen in § 2.7. Furthermore, all these subproblems interact via other span constraints. It is clear that the problem of scheduling the activities with a partial resource plan is a very hard one unless the cliques are rather small. Compare, for instance, the network in Fig. 6.6 with the one in Fig. 6.3; both refer to the same problem data, but in Fig. 6.6 the network has been drawn by fixing the resource plan only partially. This relaxed requirement has the effect of increasing the number of arcs (only by two in this case due to the small cliques) and the span lengths.

Therefore, in order to devise sensible strategies to solve the problem of scheduling the activities with a fixed partial resource plan, it may be convenient to exploit the problem of § 3.3 as a subproblem to be used for a branch and bound strategy. We recall that in the problem of § 3.3 a complete resource plan is assumed, that is, the sequencing of the activities on each clique is kept fixed. TSP techniques may be proper tools to assess convenient sequences. In this sense especially the problem of scheduling activities with a complete resource plan becomes a sensible and useful model.

**3.5. Activity scheduling with a fixed number of resource units.** In certain practical applications only the global number of available resource units is prescribed in advance and in general any partition $Q(R)$ is accepted as long as it makes the schedule feasible.

Unfortunately the requirement on the use of the resources is so relaxed in this problem that it is not possible to model it within the framework of span constraints as we did for the problems in the previous sections. It is certainly out of question to check all possible partitions $Q(R)$ until feasibility is found. Nevertheless the following considerations can provide some help.

First of all note that if $|R| = 1$ for all resources then only the trivial partition $Q(R) = \{\Gamma(R)\}$ has to be considered, with the obvious value $m(R) = 1$. In this case the problem is exactly the one of scheduling the activities with a fixed partial resource plan, which can be solved through PESP.

If on the contrary there are several resource units of each resource, then a favourable situation is faced if $s(\gamma_i, \gamma_j) = 0$ for all $i$ and $j$. In fact let us suppose that an activity schedule is given and that the goal is the minimization of the needed resource units. Let us consider the following two alternatives: either we apply the results of § 3.2 and solve an assignment problem or we add the extra requirement of having an assignment corresponding to a cyclic permutation, which amounts to solving a TSP. According to [16 p. 99], the difference between the two optimal values for this particular set of costs (in fact a graded matrix of 0's and 1's) is at most 1.

Therefore if we want to find a feasible activity schedule with a fixed number $m$ of available resource units of some resource $R$, we may consider the following approach:

1) find an activity schedule with the following fixed partial resource plan

$$Q(R) = \{\Gamma(R)\} \qquad m(R) = m.$$

If there exists a feasible solution, stop, otherwise

2) find an activity schedule with the following fixed partial resource plan

$$Q(R) = \{\Gamma(R)\} \qquad m(R) = m + 1.$$

If there is no feasible solution, stop, because the original problem does not have feasible solutions either, otherwise,

3) minimize the number $\hat{m}$ of needed resource units for the activity schedule just found. If $\hat{m} \leqq m$, stop, otherwise the situation is rather hopeless.

Special techniques could be conceived to handle the last point, but this is beyond the scope of this paper.

## 4. Applications.

**4.1. Periodic Job-Shop.** Let us first recall the usual (aperiodic) Job-Shop Problem (which will be referred to in the following as JSP). There is a set $\mathcal{J}$ of *jobs*. For each job $J \in \mathcal{J}$ there is specified a set of *activities* (or *operations*) $\gamma(J, k)$, $k = 1, \cdots, q(J)$, to be executed on a certain *piece* $r(J)$, in the sequence given by the index $k$. This way, a one-to-one association between jobs and pieces is given. Each activity $\gamma$ has a certain given duration $x(\gamma)$.

There is a set $\mathcal{M}$ of *machines*. Each activity $\gamma$ is assigned to a definite machine $M(\gamma) \in \mathcal{M}$.

The problem is the one of scheduling all activities such that

1) The activities of the same job are scheduled according to the prescribed sequence;
2) The same machine executes at most one activity at a time;
3) A prescribed criterion for the completion time has to be satisfied.

In the Periodic Job-Shop Problem (PJSP) each activity has to be repeated at regular intervals of length $T$, so it has to be considered a periodic activity. Since its duration is fixed, one periodic event identifies each periodic activity. In the periodic case each job $J$ describes a set of activities to be executed on infinitely many identical pieces $\cdots, r_1(J), r_2(J), \cdots$.

The pieces $r_i(J)$ enter the job-shop at regular intervals of length $T$ and remain in it for a certain time $T(J)$, which is the same for all of them. Let $(m(J) - 1)T < T(J) \leqq m(J)T$ for a certain integer $m(J)$. Then we identify the pieces $r_i(J)$ and $r_{i+m(J)}(J)$ in order to model this problem in our framework. This corresponds to having a finite set $R(J) := \{r_1(J), \cdots, r_{m(J)}(J)\}$ of pieces. In the framework of § 3.1, $R(J)$ is a particular resource, and we consider the pieces as resource units. Through the above identification an open system is transformed into a closed (cyclic) one.

Each machine is a resource as well. In particular it is a resource consisting of a single unit.

Each set $\mathscr{R}(\gamma)$ consists of two elements, i.e.,

$$\mathscr{R}(\gamma(J,k)) = \{R(J), M(\gamma(J,k))\}$$

and obviously

$$\Gamma(R(J)) = \{\gamma(J,1), \cdots, \gamma(J,q(J))\}$$

$$\Gamma(M) = \{\gamma(J,i) \colon M = M(\gamma(J,i))\}.$$

Then the problem is scheduling the periodic activities such that:

1) The resource plan concerning the resources $R(J)$ (pieces) is *completely* specified as follows: Due to the concept of "job," the partitions $Q(R(J))$ are the trivial partitions $Q(R(J)) = \{\Gamma(R(J))\}$ and the permutations $PA(R(J))$ must be cyclic, corresponding to the sequence $\gamma(J,1), \cdots, \gamma(J,q(J))$, which are obviously assigned a priori by technological requirements. Concerning the integers $m(J)$ assigned to each job, they are not actually imposed a priori and constitute a set of design variables subject to the only obvious limitation such that

$$m(J)T \geqq \sum_k x(\gamma(J,k)) + s(\gamma(J,k), \gamma(J,k+1)).$$

At the moment they are arbitrarily specified by the designer. We shall comment on this in a few paragraphs.

2) The resource plan concerning the resources $M$ (machines) is *partially* specified as follows: Due to the fact that each machine is considered different from all other ones and hence each activity is assigned to one definite machine, the partitions $Q(M)$ have to be the trivial partitions $\{\Gamma(M)\}$ and $m(M)$ must be equal to one. The permutations $PA(M)$ must be cyclic, but they are not specified and have to be determined in order to find a feasible schedule.

3) In the PJSP there is no completion time of course. A related measure of productivity is given by the number of processed pieces leaving the job-shop in a time unit, i.e., by $|\mathscr{J}|/T$. So, if $|\mathscr{J}|$ is considered fixed and $T$ is not imposed a priori by some other constraints, we might ask for the minimal $T$ realizing a feasible schedule, a task which could be performed via a bisection procedure.

Therefore a PJSP can be formulated within the framework of the model developed in § 3 and solved via the techniques described in § 2.

Some other features of the PJSP are worth mentioning. Let $\hat{T}$ be the minimal $T$ realizing a feasible schedule. First just note that an obvious lower bound for $\hat{T}$ is given by

$$\max\left\{\max_{M\in\mathscr{M}} \sum_{\gamma\in\Gamma(M)} x(\gamma); \max_{J\in\mathscr{J}} \sum_{\gamma\in\Gamma(J)} x(\gamma)\right\}.$$

The schedule obtained with $\hat{T}$ exhibits a critical circuit in the network model of PESP in the sense that all span constraints of the circuit are satisfied exactly at the span boundaries. This situation is the exact counterpart of the critical path for the JSP.

The concept of "selection" of a machine for the JSP, that is, prescribing the precedence order of the activities on the same machine, is translated into PJSP as a sequence specification and therefore, if all machines are given a particular selection, all resource plans are completely specified and the problem of scheduling the activities is solved as described in § 3.3.

An interesting feature of the PJSP, not present in the JSP, consists in the fact that all pieces of a certain job $J$ remain in the job-shop for an amount of time given by $T(J)$. It is important to note that increasing $T(J)$ (while keeping $T$ fixed) has the effect of relaxing the sequence constraints of the problem to such an extent that for sufficiently large $T(J)$ any resource assignment can be made feasible. However, large $T(J)$ result in a larger number of pieces simultaneously present in the job-shop and therefore produce storage problems and raise the opportunity costs associated with the capital committed to the material under process.

Thus two different objectives have been identified: 1) A productivity rate associated with the period $T$ and the number of jobs $|\mathcal{J}|$; 2) Opportunity costs associated with the $T(J)$'s and therefore with the $m(J)$'s.

As a further reference on this problem see [26]. For an example reconsider Fig. 6 where two jobs can be specified by the activities $\gamma_1, \cdots, \gamma_4$ and $\gamma_5, \cdots, \gamma_7$ respectively and the three resource units $r_6$, $r_7$, $r_8$ can be reconsidered as three different machines, i.e., three different resources, each one of them consisting of a single resource unit.

An extension of the PJSP could take into account the pallet scheduling in case the motion of pieces is performed automatically through pallets. If the pallets are a scarce resource they could be considered an additional resource. Then it is just a matter of considering new activities corresponding to pallet motions between two original activities. The resulting problem could be very difficult. A more realistic approach would be to disregard the pallet problem in the first stage of scheduling and then, once the activity schedules are fixed, minimize the number of needed pallets through the bipartite matching problem of § 3.2.

**4.2. Transportation scheduling.** In transportation one typical problem is of minimizing the number of vehicles needed to implement a given time schedule. This was solved previously by Dilworth in the aperiodic case. The periodic case has also been solved via different approaches both when deadheading is allowed and when it is not [12], [13], [18]. We recall that deadheading is the transfer of a vehicle from one terminal to another one with a trip out of schedule.

The model presented in this paper provides a rather simple framework to solve the problem of minimizing the number of vehicles with no conceptual difference between allowing deadheading or not. In this model each trip is considered as an activity $\gamma$. Then we may define the setup times $s(\gamma_1, \gamma_2)$ to be the minimal time needed for a vehicle to be available for trip $\gamma_2$ after completion of trip $\gamma_1$. If deadheading is allowed $s(\gamma_1, \gamma_2)$ takes care of the actual time needed for transferring the vehicle between two distant terminals, otherwise $s(\gamma_1, \gamma_2)$ can be simply set to infinity.

As shown in § 3.2 this is a weighted assignment and this conclusion agrees with previous results [12], [13], [15], [18].

An interesting extension consists in allowing some flexibility in the time schedule within some prescribed tolerances. So it is possible to reschedule the trips according to the results of § 3.3 given a certain complete resource plan. It is beyond the scope of this paper to give methods to derive a complete resource plan in order to find the periodic trip schedule. Actually, by combining our model with other models it is possible to develop efficient heuristics. This is a matter of current research.

**4.3. Traffic light scheduling.** This application deals with the problem of scheduling the switching times of green and red lights in a system of signals controlling urban traffic.

We look for a periodic schedule with a given period $T$. This is a common practice in several applications at least for time intervals much larger than the period.

We may model the problem by considering a green time as a periodic activity $\gamma$. Therefore there are as many activities as the independent signals. For the sake of simplicity we suppose that the green light durations $x(\gamma)$ are given a priori; so one periodic event $\varepsilon(\gamma)$, say the starting of green light, identifies each periodic activity. A more complex model involving variable green light durations can be found in [25].

The periodic activities, i.e., the green times, are subject to the following constraints: two signals $(\gamma_1, \gamma_2)$ controlling the access to the same physical area cannot be green simultaneously. So, by also taking into account proper clearance times $s(\gamma_1, \gamma_2)$ and $s(\gamma_2, \gamma_1)$, the following span constraint must be obeyed by the two signals $\gamma_1$ and $\gamma_2$:

$$\tau(\varepsilon(\gamma_2))-\tau(\varepsilon(\gamma_1))\in[-x(\gamma_1)-s(\gamma_1, \gamma_2), x(\gamma_2)+s(\gamma_2, \gamma_1)]_T.$$

Besides this type of constraint we may also model a coordination between signals controlling the same traffic flow, by imposing that the switching of a downstream signal must occur within a given time interval $[\delta^-, \delta^+]$ from the switching of the corresponding upstream signal. So PESP is enough to model this problem.

**5. Conclusions.** A general framework to deal with a large class of scheduling problems in a periodic environment has been proposed.

First the case involving only separate periodic events with constraints involving the relative position of pairs of them has been considered. It has been shown to be NP-complete and some other properties have been settled. An algorithm for it has been devised, which exhibits a satisfactory average performance in practical cases. Some extensions have also been encompassed, such as multiperiod situations.

Then the concepts and methods for periodic event scheduling problems have been exploited to deal with periodic activities using resources. Some different classes of problems have been classified and the relationships among them have been analyzed in a unitary approach.

Finally, some specific applications have been considered in detail. In the first one a periodic version of the well-known Job-Shop Scheduling problem has been treated. The second one dealt with vehicles scheduled to meet a given time table, possibly with some assigned tolerances. The last one considered setting traffic lights controlling urban traffic while complying with simple safety requirements. Each of the above problems has been modelled as a specific periodic activity scheduling problem with resources.

In conclusion, it may be pointed out that this should not be intended as a conclusive work on periodic scheduling issues. Rather, it should stimulate further research effort on such topics. For instance it is worth mentioning the possibility of refining the algorithmic aspects, for instance adapting them to specific problem features in order to achieve an improved computational efficiency. Another challenging subject contemplates optimization problems instead of only feasibility ones.

## REFERENCES

[1] K. R. BAKER, *Scheduling a full-time workforce to meet cyclic staffing requirements*, Management Sci., 20 (1974), pp. 1561–1568.

[2] J. J. BARTHOLDI III, *A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering*, Oper. Res., 29 (1981), pp. 501–510.

[3] J. J. BARTHOLDI III, J. B. ORLIN, AND H. D. RATLIFF, *Cyclic scheduling via integer programs with circular ones*, Oper. Res., 28 (1980), pp. 1074–1085.

[4] T. E. BARTLETT, *An algorithm for the minimum number of transport units to maintain a fixed schedule*, Naval Res. Logist. Quart., 4 (1957), pp. 139–149.

[5] S. E. BECHTOLD, *Work-force scheduling for arbitrary cyclic demands*, J. Operations Management, 1 (1981), pp. 205–214.

[6] R. E. BURKARD, *Optimal schedules for periodically recurring events*, Discrete Appl. Math., 15 (1986), pp. 167–180.

[7] A. CEDER AND H. I. STERN, *Deficit function bus scheduling with dead-heading trip insertions for fleet size reductions*, Transpn. Sci., 15 (1981), pp. 338–363.

[8] W. DAUSCHA, H. D. MODROW, AND A. NEUMANN, *On cyclic sequence types for constructing cyclic schedules*, Zeit. f. Oper. Res., 29 (1985), pp. 1–30.

[9] H. EMMONS, *Work-force scheduling with cyclic requirements and constraints on days off, weekends off, and work stretch*, IIE Trans., 17 (1985), pp. 8–16.

[10] S. FRENCH, *Sequencing and Scheduling*, Ellis Horwood, Chichester, United Kingdom, 1982.

[11] M. R. GAREY, D. S. JOHNSON, G. L. MILLER, AND C. H. PAPADIMITRIOU, *The complexity of coloring circular arcs and chords*, SIAM J. Algebraic Discrete Methods, 1 (1980), pp. 216–227.

[12] I. GERTSBAKH AND Y. GUREVICH, *Constructing an optimal fleet for a transportation schedule*, Transpn. Sci., 11 (1977), pp. 20–36.

[13] ———, *Homogeneous optimal fleet*, Transportation Res., 16B (1982), pp. 459–470.

[14] W-L. HSU, *On the general feasibility test of scheduling lot sizes for several products on one machine*, Management Sci., 29 (1983), pp. 93–105.

[15] V. B. KATS, *An exact optimal cyclic scheduling algorithm for multioperator service of a production line*, Automat. Remote Control, 43 (1982), pp. 538–542.

[16] E. L. LAWLER, J. K. LENSTRA, A. H. G. RINNOOY KAN, AND D. B. SHMOYS, eds., *The Traveling Salesman Problem*, John Wiley, Chichester, United Kingdom, 1985.

[17] M. O'HEIGEARTAIGH, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, eds., *Combinatorial Optimization: Annotated Bibliographies*, John Wiley, Chichester, United Kingdom, 1985.

[18] J. B. ORLIN, *Minimizing the number of vehicles to meet a fixed periodic schedule: an application of periodic posets*, Oper. Res., 30 (1982), pp. 760–776.

[19] J. B. ORLIN, M. A. BONUCCELLI, AND D. P. BOVET, *An $O(n^2)$ algorithm for coloring proper circular arc graphs*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 88–93.

[20] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[21] K. S. PARK AND D. K. YUN, *Optimal scheduling of periodic activities*, Oper. Res., 33 (1985), pp. 690–695.

[22] A. H. G. RINNOOY KAN, *Machine Scheduling Problems*, Martinus Nijhoff, The Hague, 1976.

[23] R. T. ROCKAFELLAR, *Network flows and monotropic optimization*, John Wiley, New York, N.Y., 1984.

[24] P. SERAFINI AND W. UKOVICH, *An approach towards solving periodic scheduling problems*, Ricerca Operativa, 35 (1985), pp. 17–39.

[25] ———, *A mathematical model for the fixed-time traffic control problem*, European J. Oper. Res., 41 (1989), pp. 1–14.

[26] ———, *Decomposing production scheduling problems in a periodic framework*, in Proceedings IXTH International Conference on Production Research, Cincinnati, OH, August 1987.

[27] H. S. STONE AND P. SIPALA, *The average complexity of depth-first search with backtracking and cutoff*, IBM J. Res. Develop., 30 (1986), pp. 242–258.

[28] A. TUCKER, *Coloring a family of circular arcs*, SIAM J. Appl. Math., 29 (1975), pp. 493–552.

[29] R. R. VEMUGANTI, *On the feasibility of scheduling lot sizes for two products on one machine*, Management Sci., 24 (1978), pp. 1668–1673.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 582–589, November 1989

# BIRIGIDITY IN THE PLANE*

BRIGITTE SERVATIUS†

**Abstract.** The two-dimensional generic rigidity matroid $R(G)$ of a graph $G$ is considered. The notions of vertex and edge birigidity are introduced. It is proved that vertex birigidity of $G$ implies the connectivity of $R(G)$ and that the connectivity of $R(G)$ implies the edge birigidity of $G$. These implications are not equivalences.

A class of minimal vertex birigid graphs is exhibited and used to show that $R(G)$ is not representable over any finite field.

**Key words.** generic rigidity, matroid theory

**AMS(MOS) subject classifications.** 05B35, 05C40

**1. Introduction and basic definitions.** Let $G = (V, E)$ be a simple graph on the edge set $E$, vertex set $V$. We define the *support* $\sigma(F)$ of a subset $F$ of $E$ to be the set endpoints of edges in $F$.

We define a subset $F$ of $E$ to be *independent* if $|F'| \leqq 2|\sigma(F')| - 3$ holds for all subsets $F'$ of $F$. It is well known, (see [1] and [3]), that these independent edge sets are the independent sets of a matroid, the so-called *two-dimensional generic rigidity matroid*, $R(G)$, of the graph $G$. The closure operator and rank function of this matroid will be denoted by $c$ and $r$, respectively. The term *circuit* will always refer to a circuit in $R(G)$. Some properties of circuits are discussed in [4]. Note that $R(G)$ may be considered as a restriction of the rigidity matroid of a sufficiently large complete graph.

$G = (V, E)$ is called *rigid* if $r(E) = 2n - 3$, where $|V| = n$. $G$ is called *edge birigid*, if $r(E - e) = 2n - 3$ for every $e \in E$. $G$ is called *vertex birigid*, if $G$ is rigid and

$$r(E - \text{star}(v)) = 2(n-1) - 3 = 2n - 5$$

for every $v \in V$, where star$(v)$ denotes the set of edges adjacent to $v$. We will henceforth abbreviate $E - \text{star}(v)$ with $E - v$. To simplify notation and language we will not distinguish between sets of edges and the subgraphs they induce. Some simple examples of graphs with specified rigidity properties are given in Fig. 1.

The following observations are immediate consequences of the definitions. The union of two graphs $G_1$ and $G_2$ having at most one vertex in common is not rigid, and $c(G_1 U G_2) = c(G_1) U c(G_2)$. If two rigid graphs intersect in two or more vertices, their union is rigid.

Let us call two edges of $G$ *related* if they are both contained in a rigid subgraph of $G$. Clearly the so defined relation is symmetric. An edge constitutes a rigid subgraph, which shows reflexivity. For transitivity, if edges $e$ and $f$ are contained in a rigid subgraph $H_1$ of $G$ and $f$ and $h$ are contained in a rigid subgraph $H_2$ of $G$, then $H_1$ and $H_2$ intersect in at least two vertices, namely the endpoints of $f$, so their union is a rigid graph containing $e$ and $h$. Thus rigidity induces an equivalence relation the edge set of $G$. The equivalence classes are called $r$-components. It follows that $r$-components have at most one vertex in common and that birigid graphs are at least 3-connected. Moreover, $R(G)$ can be written as the direct sum over the $r$-components of $G$. This follows from the observation that circuits are rigid, in fact edge birigid, (see [4]).

We shall often use the following property of $R(G)$: Assume the edge set $F$ induces a subgraph of $G$ containing a vertex $v$ of valence three. Then $F$ is independent if and

* Received by the editors September 12, 1988; accepted for publication (in revised form) April 12, 1989.

† Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, Massachusetts 01609.

only if there is an edge $e$ connecting neighbors of $v$ such that $e$ is not contained in $F$, and $F - v + e$ is independent. We say $R(G)$ satisfies the 1-*extendability property*, see [1]. Note that $e$ need not be contained in $G$.

If the vertices of $G$ are "generically" embedded in the plane, see [3], and the edges of $G$ are replaced by rigid bars, which are pin-jointed at their endpoints, the resulting structure will be rigid if and only if $G$ is rigid in the sense defined above. (see [2]).

If the vertices of $G$ are restricted to a line, and the edges of $G$ are again replaced by rigid bars, the resulting structure will be rigid if and only if $G$ is connected, and we may characterize the one-dimensional generic rigidity matroid $M(G)$ of the graph $G$ as follows: A subset $F$ of $E$ is independent if and only if $|F'| \leqq |\sigma(F')| - 1$ holds for all subsets $F'$ of $F$, i.e., the independent sets in this matroid are simply the edge sets of subforests of $G$. $M(G)$ is called the cycle matroid $M(G)$ of $G$, (see [6]).

Observe that $M(G)$ and $R(G)$ are matroids defined on the edge set of $G$, and that the vertex set of $G$ is used only via the support function to define independent sets. Consequently, there is no property of $M(G)$ or $R(G)$ that corresponds directly to the connectivity of $G$. Whitney [7] calls a matroid $M$ on $S$ *connected* if $r(A) + r(S - A) > r(S)$ holds for every nonempty proper subset $A$ of $S$. With this definition $M(G)$ is connected if and only if $G$ is biconnected. It is natural to ask for relations between the connectivity of $R(G)$ and the rigidity of $G$. This will be done in § 2.

Every pair of edges in a biconnected graph is contained in a cycle. A cycle is an edge-minimal vertex-biconnected graph. Note that any cycle has exactly one edge more than it needs to be connected. A biconnected graph can simply be thought of as a union of sufficiently intersecting cycles.

It is natural to look for a rigid analogue: Given a birigid graph, we can write it as a union of birigid graphs of minimal excess, where the *excess* of a rigid graph $G = (V, E)$ is defined to be $|E| - r(E)$. Observe that the only birigid graph of excess one is the complete graph on four vertices, since the average valence in a birigid graph of excess

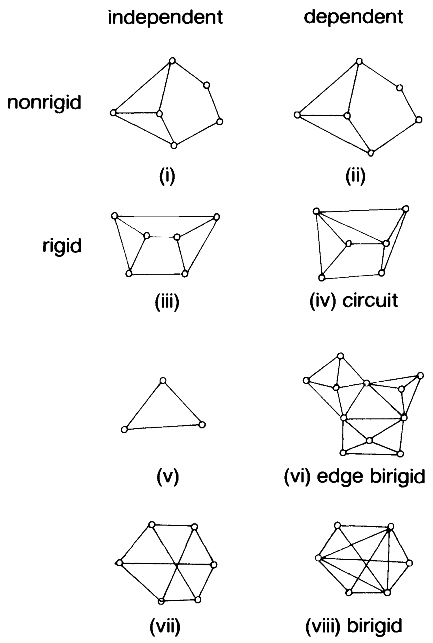


FIG. 1

one on $n$ vertices is greater than or equal to $4 - (4/n)$. Therefore, a birigid graph on more than four vertices contains a vertex of valence at least four. The removal of a vertex of valence four decreases the excess by two, therefore a birigid graph on more than four vertices has to have excess at least two. In § 3 we show that there are infinitely many birigid graphs of excess two. We give an inductive procedure to construct them all. We also show that they do not, unfortunately, fulfill the role of universal building blocks of birigid graphs.

## 2. Birigidity of $G$ and connectivity of $R(G)$.

THEOREM 2.1. *If $G$ has no isolated vertices and more than one edge, and $R(G)$ is connected, then $G$ is edge birigid, but not conversely.*

*Proof.* $G = (V, E)$ is rigid, otherwise $R(G)$ could be written as the direct sum over the rigid components of $G$. Hence $r(E) = 2|V| - 3$.

Assume that there is an edge, $e$, such that $G - e$ is not rigid. Then $r(E - e) = 2|V| - 4$ and $r(E - e) + r(e) = r(E)$. The last equation contradicts the connectivity of $R(G)$.

The converse is not true:

Let $G_o$ be minimally rigid, having $n_o$ vertices and $2n_o - 3$ edges. We attach to each edge $e_i$ a circuit $C_i$, $1 \leqq i \leqq (2n_o - 3)$, $C_i$ having $n_i$ vertices, by identifying one edge of each $C_i$ with one edge of $G_o$. Then the resulting graph is clearly rigid and hence has rank $2n - 3$, where $n = n_o + \sum_{i=1}^{2n_o-3} (n_i - 2)$. So $\sum n_i = n + 3n_o - 6$. The rank if each $C_i$ is $2n_i - 3$.

If we sum over the ranks, we get

$$\sum_{i=1}^{2n_o-3} r(C_i) = \sum_{i=1}^{2n_o-3} (2n_i - 3) = -3(2n_o - 3) + 2 \sum_{i=1}^{2n_o-3} n_i$$
$$= -6n_o + 9 + 2n + 6n_o - 12 = 2n - 3 = r(G).$$

So $M(G)$ is not connected. On the other hand, $G$ is clearly edge birigid. An example with $n_0 = 3$ is drawn in Fig. 1(vi). □

THEOREM 2.2. *If $G = (V, E)$ is birigid and $|V| > 3$, then $R(G)$ is connected but not conversely.*

*Proof.* Assume that $G$ is birigid and that $R(G)$ is not connected. Consider the connected components $R_i$ of $R(G)$. Then there is a partition of $E$,

$$E = E_1 \cup E_2 \cup \cdots \cup E_k,$$

such that

$$R(G) = R_1 + R_2 + \cdots + R_k,$$

where $R_i = R(G_i)$, with $G_i = (\sigma(E_i), E_i)$. Every $G_i$ is rigid, so it follows that

$$(1) \qquad 2|V| - 3 = r(G) = \sum_{i=1}^{k} r(G_i) = \sum_{i=1}^{k} (2|\sigma(E_i)| - 3).$$

Let $n_i$ be the number of vertices in the support of $E_i$ which are also contained in the support of some $E_j$, $i \neq j$ and let $N_i$ be the number of vertices contained only in the support of $E_i$. Denote by $N$ the number of vertices of $G$ which are contained in exactly one of the $\sigma(E_i)$'s, and by $n$ the number of vertices which occur in more than one of these supports. $N_i$, $N$, $n_i$, and $n$ satisfy the following equations:

$$\text{(i)}\ N_i = |\sigma(E_i) - \bigcup_{i \neq j} \sigma(E_j)| \qquad \text{(ii)}\ |\sigma(E_i)| = n_i + N_i$$

$$\text{(iii)}\ N = \sum_{i=1}^{k} N_i \qquad \text{(iv)}\ |V| = n + N.$$

So

$$n \leqq \frac{1}{2} \sum_{i=1}^{k} n_i. \tag{2}$$

Rewriting (1) in this new notation we obtain

$$2n+2N-3 = \sum_{i=1}^{k} (2(n_i+N_i)-3)$$

or

$$2n = 3(1-k) + \sum_{i=1}^{k} 2n_i \tag{3}$$

so that (2) and (3) give

$$\left[\sum_{i=1}^{k} 2n_i\right] - 3(k-1) \leqq \sum_{i=1}^{k} n_i,$$

or

$$\sum_{i=1}^{k} n_i \leqq 3(k-1). \tag{4}$$

Furthermore, since every cutset in a birigid graph has cardinality at least 3, we have that

$$|\sigma(E_i) \cap \bigcup_{i \neq j} \sigma(E_j)| \geqq 3,$$

which implies that $n_i \geqq 3$ for all $i$. This combined with (4) gives $3k \leqq \sum_{i=1}^{k} n_i \leqq 3(k-1)$, a contradiction.

If $R(G)$ is connected, $G$ need not be birigid: If $G$ is a wheel, $R(G)$ consists of a single circuit and hence is connected. But the removal of the center vertex leaves a nonrigid graph if the number of spokes is larger than three. □

**3. Birigid graphs of excess two.** $G$ is called *edge minimally birigid* if $G$ is birigid but $G - e$ is not birigid for all $e \in E(V)$.

In this section we will restrict our attention to an edge minimal vertex birigid graph $G = (V, E)$, which has exactly two edges more than it needs to be rigid, i.e.,

$$|E| = 2|V| - 1,$$
$$r(E) = 2|V| - 3.$$

We first list some elementary properties of $G$.

PROPOSITION 1. *Let $G$ be a birigid graph of excess two. Then*

(i) *$G$ contains at least five vertices,*

(ii) *If $e \in E(G)$, then $G - e$ is not birigid, and*

(iii) *$G$ has exactly two vertices of valence three and the remaining vertices each have valence four.*

*Proof.* (i) Simple graphs on less than five vertices do not contain enough edges to satisfy $|E| = 2|V| - 1$.

(ii) $G - e$ is not a complete graph. $G - e$ has excess one. Since the only birigid graph of excess one is $K_4$, $G - e$ is not birigid.

(iii) Since $G$ is rigid, it contains no vertex of valence less than two. Suppose that $G$ had a vertex $v$ of valence two. Let $w$ be adjacent to $v$. Then $G - w$ contains a vertex of valence one and is not rigid. Now suppose $G$ has a vertex $v$ of valence $k$. Then $G - v$ has $n - 1$ vertices and $2(n-1) - (k-1)$ edges. Since $G - v$ is rigid, $k - 1 < 4$, which implies that $k < 5$. Finally, if there are $m$ vertices of valence three, we have $3m + 4(n-m) = 2(2n-1)$, which gives $m = 2$. $\square$

The simplest birigid graph of excess two can be obtained from $K_5$ by deleting an edge. This graph contains two copies of $K_4$ as subgraphs. By "attaching" to $K_4$ two adjacent vertices of valence three, we obtain a birigid graph on six vertices. We remark that birigid graphs on more than six vertices do not contain a birigid subgraph of positive excess.

Next, we examine the circuit structure of $R(G)$.

THEOREM 3.1. *A graph on n vertices with* $2n - 1$ *edges is birigid if and only if there is a partition of the edge set E of G,*

$$E = E_1 \cup E_2 \cup \cdots \cup E_k$$

*such that* $E - E_i$ *is a circuit in* $R(G)$ *for all i, and either*

(i) $E_i$ *is an edge for* $3 \leqq i \leqq k$ *and* $E_1$ *and* $E_2$ *are stars of two vertices of valence three, or,*

(ii) $E_i$ *is an edge for* $2 \leqq i \leqq k$ *and* $E_1$ *is the union of stars of two adjacent vertices of valence three.*

*Proof.* Assume that there exists such a partition. Consider a class containing exactly one element $e$. Then $E - e$ is a circuit of $R(G)$, so $G - e$ is a graph with minimum valence at least three, and $e$ has endpoints of valence at least four in $G$. Condition (i) or (ii) imply that $G$ has two vertices of valence three and we conclude by a simple counting argument that all other vertices are of valence four.

Depending on whether or not the two vertices of valence three are adjacent in $G$, conditions (i) or (ii) imply that the removal of a vertex of valence three of $G$ results in a circuit or in a circuit with a vertex of valence two attached, a rigid graph in both cases.

Consider a vertex $v$ of valence four in $G$. Remove an edge $e$ of star$(v)$ with endpoints of valence four. $E - e$ is a circuit by assumption, and $v$ has valence three in this circuit. Recall that a circuit is edge birigid. By deleting an edge in star $v$, we therefore obtain a rigid graph in which $v$ has valence two. The removal of a vertex of valence two does not destroy the rigidity of a graph, so $E - v$ is rigid.

Conversely, assume that $G$ is edge birigid on $n$ vertices and $2n - 1$ edges. Since $r(E) = |E| - 2$, and every edge is contained in a circuit, $E$ is the union of two distinct circuits, and can be partitioned into a collection of sets $\{E_i\}$ such that $E - E_i$ is a circuit for each $i$, and $|E - E_i| = 2|\sigma(E - E_i)| - 2$, (see for example [5] or [6]). Subtracting this equation from $|E| = 2|\sigma(E)| - 1$ gives

$$(*) \qquad |E_i| = 2|\sigma(E) - \sigma(E - E_i)| + 1.$$

If $E$ and $E - E_i$ have the same support then $E_i$ is a single edge. If $\sigma(E) - \sigma(E - E_i) = 1$, then $E_i$ contains all edges of the star of a vertex in $G$. The equation $*$ gives $|E_i| = 3$. Since every vertex in $G$ has valence at least three, $E_i$ must be a star of a vertex of valence three, and the two vertices of valence three in $G$ are not adjacent because $E - E_i$ is a circuit.

If $|\sigma(E) - \sigma(E - E_i)| = 2$, then $E_i$ contains all edges of the stars of two vertices of $G$. The equation $(*)$ gives $|E_i| = 5$, so $E_i$ must be the union of two adjacent vertices of valence three in $G$.

If $|\sigma(E) - \sigma(E - E_i)| > 2$, then $E_i$ contains all edges of the star of three vertices of $G$. One of these must be of valence four. But the removal of a vertex of valence four leaves an independent set since $G$ is birigid. The desired partition is so established, and the proof of the theorem is complete. □

Examples of graphs with a partition of type (i) and (ii) are given in Fig. 2.

Clearly we can "string together" as many triangles as we wish to obtain birigid graphs of excess two of arbitrarily large size. Also the number of classes in the partition described in Theorem 1 is unbounded. From a theorem of Tutte [5], we know that, if $M$ is a matroid representable over a finite field $k$ of order $n$, and $S$ is the union of two cycles of $M$ with $r(S) = |S| - 2$, and $S_1, \cdots, S_m$ is a partition of $S$ such that $S - S_i$ is a cycle of $M$, then $m$ is bounded by $n + 1$. Hence, we have proved the following corollary.

COROLLARY. *There is no finite field $k$ such that $R(G)$ is representable over $k$ for all $G$.*

Consider an edge minimal birigid graph $G = (V, E)$. For every edge $e$ in $E$ there exists a nonempty set $V_e$ of vertices of $G$ such that $E - e - v$ is nonrigid for all $v \in V_e$. Elements of $V_e$ are called *essential vertices* for the edge $e$.

From a given birigid graph of excess two, we want to construct a larger birigid graph of excess two by attaching a vertex of valence three and removing one edge from the given graph. To formalize this idea, we introduce some notation.

Let $T$ be a graph on four vertices and three edges, where one vertex is of valence three, and construct a graph $G + T$ by identifying vertices $a, b, c$ of $G$ with the vertices of valence one in $T$.

We can now prove the following theorem.

THEOREM 3.2. *Let $G$ be a birigid graph of excess two, and let $T$ and $\{a, b, c\}$ be as described above. Then*:

(1) *$G + T$ is birigid*;

(2) *a necessary and sufficient condition for $G + T$ to be edge minimally birigid is that the set $\{a, b, c\}$ not be contained in $V - V_e$ for any edge $e$ of $G$*;

(3) *if $G + T$ is not edge minimally birigid, then there is an edge $e$ such that $G + T - e$ is birigid of excess two*; *and*

(4) *there is always a choice of $\{a, b, c\}$ such that $G - T$ is not edge minimal.*

*Proof.* (1) The removal of $T$ results in a birigid, and hence rigid graph, and the removal of any vertex $v \in G$ from $G + T$ removes at most one edge from $T$, and since $G - v$ is rigid, so is $G + T - v$.

(2) *Sufficiency.* Let $e$ be any edge of $G$. Since the intersection of $V_e$ with $\{a, b, c\}$ is nonempty, the removal of $e$ and any vertex $v$ in this intersection leaves a nonrigid graph, $G - e - v$, which has the same rigidity properties as $G + T - e - v$.

*Necessity.* Assume the existence of an edge $e$ of $G$ such that $\{a, b, c\}$ is contained in $V - V_e$. Observe that all vertices of valence four of $G$ which are not endpoints of $e$ are elements of $V_e$. Therefore, at least one vertex in the set $\{a, b, c\}$ is an endpoint of $e$.

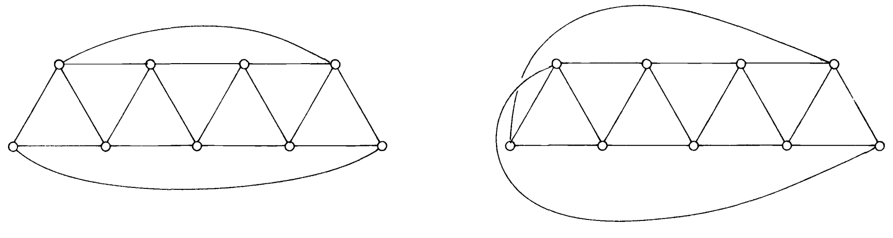


FIG. 2

There are two cases.

*Case* (a). $a$ and $b$ are endpoints of $e$, and $c$ is of valence three in $G$. Theorem 3.1 implies that a vertex $v$ of valence three is essential for a nonempty set of edges only if the two vertices of valence three in $G$ are adjacent. In this case $v$ is essential for the two edges not contained in a circuit in $G - v$. It follows that $c$ is not adjacent to a possible endpoint of $e$ of valence three and all essential vertices for $e$ are of valence four. This means that $E - v$ is rigid of zero excess, i.e., independent for all $v \in V_e$. Therefore, $E - v - e$ is independent and $e$ is not in its closure. By the 1-extendability property, $E + T - e - v$ is independent and hence rigid.

*Case* (b). $e$ has endpoints of valence four in $G$, one of them being $a$, and $b$ and $c$ are of valence three. Remove $e$ and star$(v)$ for some $v \in V_e$. Repeating the argument in (a) we show that $E - e - v$ is independent and nonrigid. Consider the $r$-components of $E - e - v$, and assume that $a$, $b$, and $c$ are contained in the same $r$-component. This component is independent, and we count that exactly three edges of $G - e$ are incident with it, contradicting the fact that $G - e$ is a circuit by Theorem 1. So, $\{a, b, c\}$ is not contained in one $r$-component of $E - e - v$, and the 1-extendability property implies that $E + T - e - v$ is independent and hence rigid for all $v$ in $V_e$, so $G + T - e$ is birigid.

(3) If $G + T$ is not edge minimally birigid, then there is an edge $e$ in $G + T$ such that $G + T - e$ is birigid. $G + T - e$ has excess two.

(4) For an edge $e$ with endpoints $a$ and $b$, both of valence four, a vertex $c$ of valence three is not essential by Theorem 1.

The proof of the theorem is now complete. □

Given an edge minimal vertex birigid graph of excess two on $n$ vertices, we can obtain an edge minimal vertex birigid graph on $n + 1$ vertices by choosing an edge $e$ in $G$ with $|V - V_e| \geqq 3$ and forming $(G - e) + T$ by identifying three vertices of $V - V_e$ with the endpoints of $T$ of valence one. In fact, we obtain all birigid graphs of excess two by this process.

THEOREM 3.3. *Let $G$ be a birigid graph of excess two with $|V| > 5$. Let $v$ be one of its vertices of valence three,* $T = \text{star}(v)$ *and let $x$, $y$, and $z$ denote the vertices adjacent to $v$. Then there is an edge $e$ with endpoints in $\{x, y, z\}$ such that $e$ is not an edge of $G$ and $G - T + e$ is birigid.*

*Proof.* $|V| > 5$ insures that $G - T$ is not complete.

There are two cases.

*Case* (a). The two vertices of valence three in $G$ are adjacent. By Theorem 1, the removal of $v$ leaves a circuit, $C$, with a vertex, $x$, of valence two attached. Assume $x$ and $y$ are in the same rigid component of $C + x - w$, where $w$ is a vertex of valence four in $C$. We count that exactly three edges leave this component, contradicting the fact that a cutset of $C$ has cardinality greater than three.

Observe that $x$ is not adjacent to $y$ or $z$; this would contradict the birigidity of $G$. So, $x$ and $y$ are never in the same rigid component of $C + x - w$, and neither are $y$ and $z$; therefore, $C + x - w + e$ is rigid if $e$ is one of $(x, y)$, $(x, z)$, respectively.

*Case* (b). The two vertices of valence three in $G$ are not adjacent. By Theorem 1, if we remove $v$, we are left with a circuit $C$. Let $w$ be a vertex of valence four in $C$. $C - w + T$ is rigid of zero excess, hence independent and, consequently, $C - w$ is independent and nonrigid. By the 1-extendability property there exists an edge $e$ with endpoints in $\{x, y, z\}$ such that $C - w + e$ is rigid. However, the choice of $e$ depends on $w$, and we have to find an $e$ that achieves rigidity independently from the choice of the removed vertex $w$.

If $C$ contains already two of the possible three edges with endpoints in $\{x, y, z\}$, we are done. Assume now that $C$ does not contain $e = (x, z)$ and $f = (y, z)$ and there is a vertex $w$ of valence four in $C$ such that $x$ and $z$ are in the same rigid component $A$

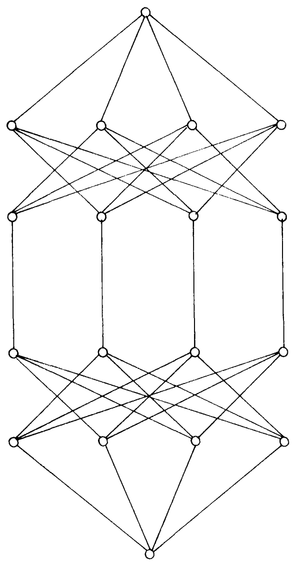


FIG. 3

of $C - w$, but $C - w + f$ is rigid and that there is a vertex $u$ of $C$ such that $y$ and $z$ are in the same rigid component $B$ of $C - u$ and $C - u + e$ is rigid. $A$ and $B$ intersect in at least one edge, since $z$ is of valence three in $C$, and their union is not equal to $C$. $A$ contains at least two vertices which are not in $B$, so there are at least three edges of $A - B$ incident with vertices of $B$, and by symmetry, three edges of $B - A$ are incident with vertices of $A$. We count that exactly four edges leave each of $A$ and $B$. So $|(C - (A \cup B)| \leqq 2$, contradicting the fact that $C - (A \cup B)$ contains a vertex.

Therefore, we can always find an edge $e$ with endpoints in $\{x, y, z\}$ such that $C - w + e$ is rigid for all vertices $w$ in $C$, i.e., $G - T + e$ is birigid. $\square$

We have now found all birigid graphs of excess two, and we have seen that they are not only edge minimally birigid, but also minimal in the sense that they do not, with the exception of the ones on five and six vertices, contain any birigid subgraph of positive excess. Now we ask if every birigid graph on more than six vertices contains a birigid graph of excess two. The answer is no. The graph in Fig. 3 is birigid, has excess three, and is minimal. The question of whether or not there are minimally birigid graphs of arbitrary excess is still open.

**Acknowledgment.** I wish to thank Jack Graver for introducing me to $R(G)$, and for his guidance throughout this research.

## REFERENCES

[1] J. E. GRAVER, *A combinatorial approach to infinitesimal rigidity*, Technical Report M-10, Department of Mathematics, Syracuse University, August, 1984.

[2] G. LAMAN, *On graphs and rigidity of plane skeletal structures*, J. Engrg. Math., 4 (1970), pp. 331–340.

[3] L. LOVASZ AND Y. YEMINI, *On generic rigidity in the plane*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 91–98.

[4] K. SUGIHARA, *On redundant bracing in plane skeletal structures*, Bulletin of the Electrotechnical Laboratory, 44 (1980), pp. 78–88.

[5] W. T. TUTTE, *Lectures on matroids*, J. Res. Nat. Bur. Standards, 69B (1965), pp. 1–48.

[6] D. J. A. WELSH, *Matroid Theory*, Academic Press, London, 1976.

[7] H. WHITNEY, *On the abstract properties of linear independence*, Amer. J. Math., 57 (1935), pp. 509–533.

SIAM J. DISC. MATH.
Vol. 2, No. 4, pp. 590–599, November 1989

# PRIME TESTING FOR THE SPLIT DECOMPOSITION OF A GRAPH*

JEREMY SPINRAD†

**Abstract.** This paper develops an $O(n^2)$ algorithm for testing whether a graph is decomposable with respect to the split decomposition. The fastest previous algorithm required $\Omega(n^3)$ time for this problem. This leads to an $O(n^2)$ expected time algorithm for computing the split decomposition of a graph.

**Key words.** algorithms, graph decomposition, join decomposition, split decomposition

**AMS(MOS) subject classifications.** 68Q25, 05C

**Introduction.** The split decomposition for directed graphs, which is defined in the next section of this paper, was developed by Cunningham. This decomposition has also been called the join decomposition of a graph. A nontrivial decomposition of a graph $G$ can be used to simplify the weighted independent set problem and the dominating set problem on $G$ [Cu]. Two independent papers have been written which recognize circle graphs by decomposing a graph $G$ using the split decomposition. $G$ is a circle graph if and only if each of the separate components is a circle graph, and if there is no further decomposition possible, the representation of the component on the circle is unique [Bo2], [GHS]. This is used to develop a polynomial test to determine whether a graph is a circle graph. Bouchet's algorithm in particular relies on repeated calls to a prime testing subroutine [Bo2]. The split decomposition also plays a key role in efficient algorithms for isomorphism of circle graphs and circular-arc graphs, and simplifies algorithms for circular-arc graph recognition [Hs].

The split decomposition is a generalization of a well-known form of graph decomposition called the substitution decomposition [Mo2], [MR]. There is a natural correspondence between the cographs [CLS], which are "completely decomposable" with respect to the substitution decomposition, and the distance hereditary [BM] or completely separable [HM] graphs, which are "completely decomposable" with respect to the split decomposition. Another close correspondence exists between the permutation graphs [Go], [Sp] and the circle graphs. A permutation graph has a unique representation if it is indecomposable with respect to the substitution decomposition, while a circle graph has a unique circular representation if it is indecomposable with respect to the split decomposition.

In this paper, we show that we can determine whether a graph is prime (i.e., indecomposable) with respect to the split decomposition in $O(n^2)$ time, where $n$ is the number of vertices in $G$. Since almost all graphs are prime with respect to the split decomposition, this gives an $O(n^2)$ expected time algorithm for computing the entire split decomposition. Previous algorithms for this problem are due to Cunningham [Cu] and Bouchet [Bo1], and take $O(n^4)$ and $O(n^3)$ time, respectively.

Both Cunningham and Bouchet rely on a subroutine called Separate, which takes two edges $e_1$, $e_2$ as input and tests whether there is a split of $G$ into $V_1$, $V_2$ such that $e_1$ goes from $V_1$ to $V_2$, and $e_2$ goes from $V_2$ to $V_1$. Each call to Separate takes $O(n^2)$ time [Cu], which can be reduced to $O(m)$ for undirected graphs, where $m = |E|$ [GHS]. Cunningham's algorithm takes a spanning intree and a spanning outtree for $G$, and calls Separate for each pair of tree edges, thus taking $O(n^4)$ time. Bouchet's algorithm is more

* Received by the editors March 19, 1987; accepted for publication (in revised form) February 16, 1989.

† Department of Computer Science, Vanderbilt University, Nashville, Tennessee 37235. This research was supported by National Science Foundation grant DCR-8604577.

complicated to describe, but it chooses to make calls to Separate in a particular order, cutting the number of calls to $O(n)$. Bouchet's algorithm thus takes $O(n^3)$ time to decompose the graph. We note that these algorithms actually find the entire split decomposition, rather than simply testing whether the graph is prime. However, Cunningham's and Bouchet's algorithms make $O(n^2)$ and $O(n)$ calls to Separate, respectively, if the graph is prime, and thus do not seem to run faster when the graph has a very simple decomposition. This paper takes a different approach; we are able to test in $O(n^2)$ time whether there is a split such that two vertices $a$ and $b$ belong on the same side of the split. The advantage of this approach is that it is much easier to find a pair of vertices which belong on the same side of the split than to find a pair of edges which go across the split; any trio of vertices contains some pair which belong on the same side of the split.

**Definitions.** In this paper, for any $x$ in $V$, $N(x)$ will denote the set

$$\{y \in V : (x,y) \in E\}.$$

For a set $S$ of vertices,

$$N(S) = \{Y \in V - S : \exists s \in S, (s,y) \in E\}.$$

As in the original paper defining the split decomposition [Cu], we will assume that the input graph $G = (V, E)$ is strongly connected. Let $V_1$, $V_2$ be a partition of $V$ such that $|V_1| \geqq 2 \leqq |V_2|$. Define $V_{1in} = \{v \in V_1 :$ there exists $w \in V_2, (w, v) \in E\}$, and let $V_{1out} = \{v \in V_1 :$ there exists $w \in V_2, (v, w) \in E\}$. $V_{2in}$ and $V_{2out}$ are defined analogously. $V_1$, $V_2$ is a *split* of $G$ if for all $v \in V_{1out}$, $w \in V_{2in}$, $(v, w) \in E$, and for all $x \in V_{1in}$, $y \in V_{2out}$, $(y, x) \in E$. In Fig. 1, $\{a, b\}$, $\{c, d\}$ is a split of $G$; $V_{1out} = \{a, b\}$, $V_{1in} = \{a\}$, $V_{2out} = \{d\}$, $V_{2in} = \{c, d\}$. If $G$ has no split, $G$ is called *prime* with respect to the split decomposition.

An *outforest* is a directed forest in which each component is a rooted tree with edges directed from parent to child. An *inforest* consists of rooted trees with edges directed from child to parent. The *level number* of a vertex $v$ in an outforest is the length of the path from a root of the outforest to $v$, and the level number of a vertex $v$ in an inforest is the length of the path from $v$ to a root of the inforest. A level of an inforest or outforest is the set of vertices which have the same level number.

For a split $V_1$, $V_2$ of a graph, we can define a *marker* element $v$. The marker $v$ can be used to reconstruct the relationships between vertices from $V_1$ and vertices in $V_2$; $v$ is a new vertex that has edges to $V_{1in}$ and $V_{2in}$, and edges from $V_{1out}$ and $V_{2out}$. The split decomposition of $G$ is formed by choosing any split $V_1$, $V_2$ of $G$ with marker $v$, and recursively decomposing $V_1 \cup \{v\}$ and $V_2 \cup \{v\}$. This decomposition divides $G$ uniquely into prime subgraphs and certain "highly decomposable" subgraphs; Cunningham [Cu]

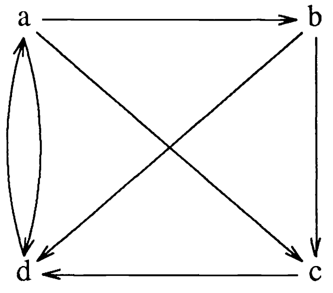


FIG. 1

gives precise, unique decomposition theorems for both the directed and undirected case of the split decomposition.

**Algorithms.** The algorithm described in this paper will start with any pair $a$, $b$ of vertices in $V$, and test whether there is a split of $G$ such that $a$ and $b$ belong on the same side of the split, which we will arbitrarily call $V_1$. In any trio of vertices $a$, $b$, $c$, there must be some pair that belongs on the same side of a split if $G$ is not prime. Therefore, to test whether $G$ is prime, we simply need to run this algorithm for the pairs $(a, b)$, $(a, c)$, and $(b, c)$.

In the algorithm for testing whether a graph is prime with respect to the split decomposition, we keep track of a number of sets. Each set is organized as a type of forest, either as a breadth first search outforest, or as a breadth first search inforest. Each vertex is in one inforest and one outforest. The first level of a forest (that is, the set of roots of the forest) is called an *initial level*.

The proof of correctness of the algorithm is based on several properties of the forests which will be maintained by all procedures of the algorithm. These properties are stated explicitly here. For these properties, we assume that there is a split of $G$ into $V_1$, $V_2$ such that $a$ and $b$ are in $V_1$.

*Property* 1. $V_2$ is contained within a single inforest and a single outforest.

*Property* 2. Let $v$ be a vertex from the initial level of an outforest $F$. Then there is a vertex $u$ that is not in $F$, such that $(u, v)$ is an edge of $G$, and for any vertex $f$ of $F$ that is not at the initial level, $(u, f)$ is not in $G$.

*Property* 3. Let $v$ be a vertex from the initial level of an inforest $F$. Then there is a vertex $u$ that is not in $F$, such that $(v, u)$ is an edge of $G$, and for any vertex $f$ of $F$ that is not at the initial level, $(f, u)$ is not in $G$.

Intuitively, each forest can be thought of as an "approximation" of $V_2$, with the initial level of an outforest being an approximation of $V_{2in}$, and the initial level of an inforest being an approximation of $V_{2out}$. The algorithm runs through a partitioning procedure, making successively finer approximations of $V_2$ when an inconsistency is found.

To quickly summarize the algorithm, once we have our initial forests, we find edges that go between different forests. Whenever such an edge $(x, y)$ is found, we split the level of the outforest that contains $y$ into neighbors and nonneighbors of $x$; the neighbors of $x$ from this level become the initial level of a new outforest. Similarly, the level of the inforest that contains $x$ is split into vertices that have an edge to $y$ and vertices that do not have an edge to $y$; the vertices that have an edge to $y$ become the initial level of a new inforest. The rest of this section provides a more detailed description of the algorithm, as well as a proof of correctness and an analysis of the time complexity of the algorithm.

A pseudoprogram for creating a first set of forests, which we will call Createforests, is shown below. Createforests begins with three sets of vertices that cannot occur together in $V_2$ and performs a variant of breadth first search from each set to place unmarked vertices in a breadth first search forest from one of these three sets.

```
Createforests(a,b,V,E);
  G1 := N(a) − N(b) − {b};
  G2 := N(b) − N(a) − {a};
  G3 := N(a) ∩ N(b);
  mark all vertices in the sets {a}, {b}, G1, G2, G3;
  for i := 1 to 3 do
    begin {create a breadth first outforest with initial level Gi}
      thisforest := ∅;
    curlevel := $\emptyset$
    nextlevel := Gi;
    while (nextlevel $\neq \emptyset$) do begin
       thisforest := thisforest $\cup$ nextlevel;
       curlevel := nextlevel;
       nextlevel := $\emptyset$;
       for each vertex v in curlevel do
          for each neighbor w of v do
             if w is unmarked then begin
                add an edge in the outforest from v to w;
                add w to nextlevel
                mark w
             end;
    end;
    {we now create an inforest for the same set of vertices}
    unmark all vertices in thisforest;
    curlevel := $\emptyset$;
    nextlevel := $\{v \in$ thisforest : $\exists(v,w), w \notin$ thisforest$\}$;
    mark all vertices in nextlevel;
    while (nextlevel $\neq \emptyset$) do begin
       curlevel := nextlevel;
       nextlevel := $\emptyset$;
       for each vertex v in curlevel do
          for each w such that w $\rightarrow$ v do
             if (w $\in$ thisforest) and (w is unmarked) then begin
                add an edge in the inforest from w to v;
                add w to nextlevel;
                mark w;
             end;
    end;
 end;
```

OBSERVATION 1. *The only routine in this algorithm that adds edges to forests is* Createforests. *Furthermore, every level of a forest $F$ at any time in the algorithm comes from a single level of the forest created by* Createforests.

OBSERVATION 2. *There is never any edge from a vertex at level $i$ of an outforest $F$ to a vertex at level greater than $i + 1$ of $F$. There is never any edge to a vertex at level $i$ of an inforest $F$ from a vertex at level greater than $i + 1$ of $F$.*

Figure 2 shows a sample graph and the inforests and outforests that come from a call Createforests $(a, b, V, E)$. We note that we also consider $a$ and $b$ by themselves to be forests. Every vertex must be in an inforest and an outforest, since the original graph is strongly connected. We will assume that edges of the forests can be traversed in either direction. The inparent of $x$ is the parent of $x$ in the inforest containing $x$, and the outparent of $x$ is the parent of $x$ in the outforest which contains $x$.

LEMMA 1. *For any split of $G$ such that $a$ and $b$ are in $V_1$, properties* 1, 2, *and* 3 *hold at the end of the call to* Createforests.

*Proof.* It is easy to see that Properties 2 and 3 hold at the end of Createforests. The initial levels of any outforest contain exactly those vertices which are neighbors of $a$ and/or $b$, so Property 2 must hold. The initial level of an inforest is defined to be the set of vertices which have edges to vertices outside the forest, so Property 3 must hold.

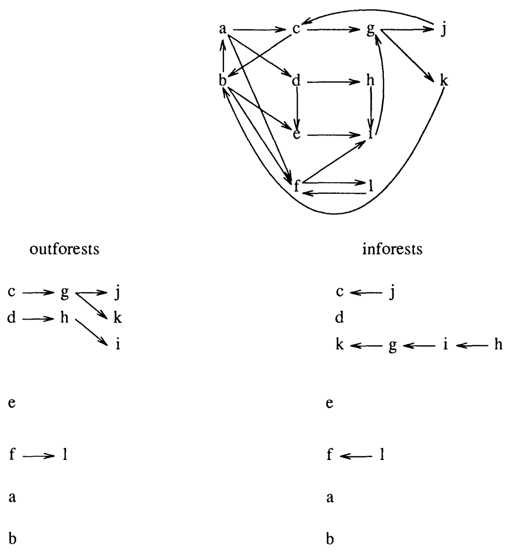


FIG. 2

We now consider Property 1, which states that $V_2$ is contained in a single inforest and a single outforest. Since each inforest was created by rearranging the vertices from an outforest, we only need to verify that $V_2$ must come from a single outforest.

Assume that there are two outforests $F_1$, $F_2$ which contain vertices from $V_2$. Let $x$ be a vertex of $V_2 \cap F_1$ that is at the first possible level of $F_1$, and let $y$ be a vertex of $V_2 \cap F_2$ that is at the first possible level of $F_2$. Both $x$ and $y$ have an inedge from some vertex in $V_1$, since all vertices before $x$ and $y$ in the outforest are in $V_1$ and $a$ and $b$ are in $V_1$. Therefore, both $x$ and $y$ must be in $V_{2in}$. However, $x$ and $y$ cannot have edges from the same set of vertices in $V_1$, since the vertex that brought $x$ into its outforest was not the same vertex that brought $y$ into its outforest. Therefore, $x$ and $y$ cannot both be in $V_{2in}$, and this is not a valid split of $G$. □

An edge $(v, w)$ such that $v$ and $w$ are in different outforests is called an *out-crossedge.* Similarly, an edge $(v, w)$ such that $v$ and $w$ are in different inforests is called an *in-crossedge.* Collectively, such edges are called crossedges. Let $(x, y)$ be a crossedge. When this crossedge is found, it will have an effect on the inforest that contains $x$ and the outforest that contains $y$. We will discuss the effects on the outforest containing $y$. Let $F$ be the outforest that contains $y$, and let $L$ be the set of vertices on the same level of $F$ as $y$.

We use $(x, y)$ to partition $F$. We move the vertices of $L \cap N(x)$ to the front of a new outforest. Vertices of $F$ which are at a higher level than $y$ are placed in the same forest as their parents, so no extra work is required to deal with these vertices. The subroutine to partition an outforest into subforests on the basis of a crossedge $(x, y)$ is given below and is called Outdivide. A similar program, which we will call Individe, is used to split the inforest which contains $x$. We note that Outdivide can be called for any pair of vertices $(x, y)$ in different outforests, whether or not there is a crossedge from $x$ to $y$.

```
Outdivide (x,y);
  F := the outforest which contains y;
  L := vertices of F on the same level as y;
  L1 := $N(x) \cap L$;
```

```
    remove L1 from F;
    create a new outforest with initial level equal to L1;
    { note that descendants of vertices in L1 become part of the new outforest }
```

LEMMA 2. *Suppose that properties* 1, 2, *and* 3 *are true before a call* Outdivide$(x, y)$. *Then Properties* 1, 2, *and* 3 *remain true after the call to* Outdivide.

*Proof.* Property 3 is not affected by the call to Outdivide, so we only need to consider Properties 1 and 2.

We first consider Property 2, which states that every vertex in the initial level of an outforest has an edge from a vertex $u$, such that $u$ does not have an edge to any vertex in a higher level of the outforest. Let $L$ be the level that contains $y$, and let $F$ be the forest that contains $y$. If $L$ is the initial level of $F$, it should be clear that Property 2 remains true for the new forest. If $L$ is not the initial level of $F$, then every vertex of $L$ has an edge from some vertex $u$ that is in the level below $L$ in $F$. Vertex $u$ cannot have an edge to any vertex at a level above $L$ in $F$, so Property 2 remains true for the new forest created by the call to Outdivide.

We now consider Property 1, which states that $V_2$ must be contained in a single inforest and a single outforest. Let $v$ be a vertex of $V_2$ from the lowest possible level of $F$, and let $w$ be a vertex of $V_2$ from the lowest possible level of $F$ that is placed in the other subforest of $F$ by the call to Outdivide. Both $v$ and $w$ must be in $V_{2in}$, since each vertex is either at the initial level of $F$, or has an edge from a parent which is in $V_1$, or has an edge from $x$. This implies that $v$ and $w$ must have the same parent, since the vertex which brought $v$ into the outforest is a member of $V_1$, and also brings $w$ into the subforest. Children of the same parent can only be placed in different forests during a call Outdivide$(x, y)$ if both are in the same level as $y$, and one has an edge from $x$ while the other does not. Since $x \in V_1$, this contradicts the fact that both $v$ and $w$ are in $V_{2in}$, so $v$ and $w$ will remain in the same forest after the call to Outdivide. □

After the Outdivide procedure has been performed, we may have new crossedges, since an edge between two vertices from the same forest $F$ may now go between two different subforests of $F$. We use the procedure below, which continues dividing the outforests until every crossedge $(u, v)$ has been used to Outdivide the outforests which contain $v$. We say that a vertex $y$ goes on $x$'s *outforest active list* as soon as $x$ and $y$ are put into different outforests. Vertex $y$ leaves $x$'s outforest active list when we divide the level $L$ that contains $y$ into neighbors and nonneighbors of $x$ during a call to Outdivide.

```
Outstabilize;
  while some outforest active list $\neq \emptyset$ do
    find a pair $(u,v)$ such that v is on u's outforest active list;
    Outdivide (u,v)
  end;
```

A similar routine, which we will call Instabilize, performs the same task for inforests. There is only one major task left to describe: the division of the outforests, which tells us that certain vertices cannot belong together in $V_2$, must also cause a division of the corresponding inforests.

We use a routine called Insubdivide to divide the inforests. Insubdivide takes as input a set of outforests $F_1, F_2, \cdots, F_k$ that are part of a single inforest $F$. Each level $L$ of $F$ is partitioned into sublevels $L_i$ such that every vertex in $L_i$ is contained in $F_i$. If a vertex $x$ is in $F_i$, but inparent$(x)$ is not in $F_i$, then $x$ is placed in the initial level of a new inforest. This procedure is described in the pseudoprogram shown below. At the end of Insubdivide, every vertex in an inforest $F$ is contained in a single outforest. This will in general create many new in-crossedges, which will be used to subdivide the inforests by a call to Instabilize.

```
Insubdivide($F_1$,$F_2$,$\cdots$,$F_i$,F);
  L := the first level of F;
  partition L into subgroups which are in the same outforest, each
    of which becomes the initial level of an inforest;
  L := the next level of F;
  while (L $\neq \emptyset$) do begin
    newforest := {v $\in$ L : v, inparent(v) are in different outforests};
    L := the next level of F;
    partition newforest into subgroups which are in the same outforest;
    remove newforest from F;
    create new inforests with initial levels equal to the subgroups of newforest;
  end;
```

LEMMA 3. *If Properties* 1, 2, *and* 3 *hold before a call to* Insubdivide, *then properties* 1, 2, *and* 3 *hold after a call to* Insubdivide.

*Proof.* Property 2 is a property of outforests, and does not need to be verified since the outforests do not change as a result of this procedure.

Property 3 says that every vertex in the initial level of an inforest $F$ has an edge to some vertex $u \in V - F$, such that $u$ does not have any edges from vertices at a higher level of $F$. Note that the subforests created by Insubdivide are formed from $F$ by moving some vertices to the initial levels of new forests. Let $x$ be a vertex in the initial level of one of the forests $F_j$ created by Insubdivide. If $x$ was in the initial level of $F$, Property 3 clearly remains true for $x$. If $x$ is at level $i$ of $F$, then every vertex of $F_j$ that is not in the initial level of $F_j$ was at level at least $i + 1$ of $F$, while the parent $p$ of $x$ is at level $i - 1$. Vertex $x$ has an edge to $p$, which must be in a different forest, since $x$ was moved to the initial level, while no vertex at a higher level of $F_j$ has an edge to $p$.

Property 1 states that $V_2$ must be contained in a single inforest and a single outforest. Let $v$ be a vertex of $V_2$ from the lowest possible level of $F$, and let $w$ be a vertex of $V_2$ from the lowest possible level of $F$ which is placed in a different inforest from $v$ during a call to Insubdivide. Since Property 2 holds before the call to Insubdivide, $v$ and $w$ must be in the same outforest. Both $v$ and $w$ must be in $V_{2out}$, since each vertex is either at the initial level of $F$ or has an edge to its parent, which is a member of $V_1$. This implies that $v$ and $w$ must have the same parent, since the vertex that brings $v$ into the inforest is a member of $V_1$ and also brings $w$ into the inforest. Since $v$ and $w$ have the same parent and were in the same outforest before the call to Insubdivide, $v$ and $w$ remain in the same forest after the call to Insubdivide. □

The program below, called Primetest, is used to determine whether there is any split of $G$ such that $a$ and $b$ are both in $V_1$ and can easily be used as a subroutine to determine whether $G$ is prime. When Primetest halts, no outforest subdivides any inforest, and no inforest subdivides any outforest. Therefore, after Primetest halts, the vertices of each inforest correspond to the vertices of one outforest. We will therefore say that the vertices are divided into forests, since we no longer have to distinguish between inforests and outforests.

```
Primetest (a,b,G);
  Consider all vertices, including a and b, as part of a single forest;
  Createforests (a,b,V,E);
  while (some inforest or outforest has been split in the last phase) do
```

```
    begin
      Outstabilize;
      for each inforest F which is in > 1 outforest $F_1 \cdots F_i$ do
        Insubdivide $(F_1, \cdots, F_i, F)$;
      Instabilize;
      for each outforest F which is in > 1 inforest $F_1 \cdots F_i$ do
        Outsubdivide $(F_1, \cdots, F_i, F)$;
    end;
  if (some forest contains > 1 vertex) then return ('split possible')
                                     else return ('no such split');
```

THEOREM 1. Primetest $(a, b, G)$ *answers that a split is possible if and only if there is a split of G such that a and b are in* $V_1$.

*Proof.* Suppose that Primetest answers that a split is possible. There must be some forest $F$ that has at least two vertices at the end of Primetest. The split of $G$ is $V - F$, $F$. $V_{2in}$ is the initial level of the outforest corresponding to $F$, while $V_{2out}$ is the initial level of the inforest corresponding to $F$. No vertex $v_1$ of $V_1$ has an edge to any vertex $v_2$ of $V_2 - V_{2in}$; $v_2$ must have been removed from $v_1$'s outforest active list, and if $(v_1, v_2)$ was an edge, this would have placed $v_2$ at the initial level of an outforest. Similarly, no vertex of $V_1$ can have an edge from $V_2 - V_{2out}$. Finally, any vertex $v_1 \in V_1$ which has an edge to (from) a subset of $V_{2in}$, $(V_{2out})$ must have an edge to every vertex in that set, or the level of the outforest (inforest) would be partitioned into the neighbors and nonneighbors of $v_1$.

Suppose that Primetest answers that no split is possible, but $G$ has a split $V_1$, $V_2$. $V_2$ must contain members of at least two different forests, since $|V_2| \geqq 2$ in any split. Lemma 1 tells us that $V_2$ was contained in a single inforest and a single outforest after the call to Createforests. Lemma 2 tells us that $V_2$ could not have been divided into separate forests during a call to Outdivide, and Lemma 3 tells us that $V_2$ could not have been divided into separate forests during a call to Insubdivide. The proofs that $V_2$ cannot be divided into different forests during calls to Individe and Outsubdivide are very similar to the proofs of Lemmas 2 and 3 and will be left to the reader. Since separation of vertices into subforests can only happen within these routines, $V_2$ must be contained in a single forest at the end of the call to Primetest. □

Createforests is similar to breadth first search, and runs in $O(n + m)$ time, where $m = |E|$. Each vertex enters each other vertex's inforest active list once, so the total number of checks during Instabilize (Outstabilize) is $O(n^2)$. In Individe and Outdivide, the adjacency between any pair of vertices is examined at most twice (when the level that contains one vertex is partitioned into neighbors and nonneighbors of the other vertex), so the total number of times the adjacency matrix is examined is $O(n^2)$.

Within the procedures Individe and Outdivide, we divide the level $L$ of a forest containing a vertex $y$ depending on relationships to a vertex $x$. All vertices in $L$ are then removed from an "active list" for $x$ and never return to the active list. We want to divide $L$ using time proportional to the number of vertices in $L$, which would give an $O(n^2)$ bound for the time used for dividing levels throughout the algorithm. It is easy to locate the position of $y$ in constant time by maintaining pointers to the position of $y$ in the set of inforests and the set of outforests. However, since the forest is changing, it may not be obvious how we can efficiently find other vertices in this level. After a forest is subdivided, we traverse each subforest and link vertices at the same level in a doubly linked list. The traversal takes $O(n)$ time, and the total number of times forests are subdivided is $O(n)$, so maintaining links between vertices at the same level takes $O(n^2)$ time. Using these lists, it is easy to see that the total time spent within the procedures Individe and Outdivide is $O(n^2)$.

The final routines which need to be analyzed are Insubdivide and Outsubdivide. These involve traversing a forest and placing each vertex in the appropriate subforest. If we assume that we know the name of the outforest and inforest containing $x$, it is easy to see that the routine takes $O(n)$ time per call. Since these routines are called only when a forest is divided, this gives a time bound of $O(n^2)$ spent within Insubdivide and Outsubdivide. We store the name of the inforest and outforest that contains vertex $x$ in position $x$ of an array. Whenever a forest is subdivided, we give a number to each subforest. We traverse each subforest and change the appropriate values in the array. Once again, the time taken is $O(n)$ each time a forest is subdivided, and thus the total time needed to maintain the array of names is $O(n^2)$.

For the special case of undirected graphs, the algorithm becomes somewhat simpler. There is only one type of forest, so there is no need for the procedures Insubdivide and Outsubdivide. On sparse undirected graphs, it is possible to modify the procedures Outdivide and Outstabilize to create an $O(m \log n)$ algorithm for prime testing; it is not clear that this can be done for directed graphs. Since the details of the $O(m \log n)$ algorithm require quite a bit of modification to the existing data structures, we give only a short sketch here.

In the new implementation, instead of calling a routine Divide $(x, y)$ to deal with a crossedge $(x, y)$, we use a routine Splitall$(x)$ to divide each forest which does not contain $x$. Splitall$(x)$ marks each vertex on $x$'s adjacency list, partitions any levels of forests that contain a neighbor of $x$, and makes the vertices that are neighbors of $x$ initial levels of new forests. This splitting can be accomplished in $O(|N(x)|)$ time in a careful implementation.

Splitall is called by a routine corresponding to Instabilize. Every crossedge in an undirected graph has at least one endpoint in an initial level, which makes it possible to restrict our calls of Splitall$(x)$ to vertices $x$ which are in initial levels. A vertex $x$ in an initial level can be used for a call Splitall$(x)$ if either Splitall$(x)$ has never been called, or the current size of $x$'s level is at most one half the size of $x$'s group during the last call to Splitall$(x)$. This guarantees that Splitall$(x)$ is not called more than $\log n + 1$ times. If no vertex is eligible for a call to Splitall$(x)$, let $F$ be the forest with the largest initial level. Every vertex in an initial level of a forest other than $F$ has attempted to split $F$, so $V - F$, $F$ is a split of $G$.

**Computing the entire decomposition.** The entire split decomposition can be found by repeatedly testing whether the graph is prime; if it is not, the graph is decomposed into $V_1 \cup \{v\}$, $V_2 \cup \{v\}$, where $v$ is a new vertex such that $(v, w)$ is an edge if and only if $w \in V_{1in}$ or $w \in V_{2in}$, and $(w, v)$ is an edge if and only if $w \in V_{2out}$ or $w \in V_{1out}$. We then attempt to decompose each of the two pieces of $G$. Since there are at most $n$ decompositions before each component becomes prime, this takes $O(n^3)$ time in the worst case.

In this section, we show that if every labeled graph on $n$ vertices is considered equally likely, then the expected running time for computing the split decomposition of a graph in this manner is $O(n^2)$. The $O(n^2)$ expected behavior follows from the fact that almost every graph is prime as $n$ becomes large. Möhring has previously shown this to be true for the substitution, or modular, decomposition, which is a special case of the split decomposition. Möhring and Radermacher [MR] have observed that the same techniques can be used for the split decomposition; the proof is included here to make the paper self-contained. We note that Bouchet's algorithm [Bo1], which also takes $O(n^3)$ time in the worst case, has $\Omega(n^3)$ expected behavior; on a prime graph, Bouchet's algorithm will make $n$ "local complement" steps, each of which takes $\Omega(n^2)$ time in the average case.

The number of graphs on $k$ vertices such that $|V_1| = j$ is at most

$$\binom{k}{j} * 2^{j(j-1)} * 2^{(k-j)(k-j-1)} * \sum_{i=0}^{j} \binom{j}{i} * \sum_{i=0}^{j} \binom{j}{i} * \sum_{i=0}^{k-j} \binom{k-j}{i} * \sum_{i=0}^{k-j} \binom{k-j}{i}.$$

The formula above calculates the number of ways to choose $V_1$, and multiplies this by the number of subgraphs on $V_1$ and $V_2$, times the number of ways to choose $V_{1in}$, $V_{1out}$, $V_{2in}$, and $V_{2out}$. This formula can be simplified to $\binom{k}{j} * 2^{2j^2 + k^2 - 2jk + k}$. If we let $j$ range from 2 to $k - 2$, $2^{2j^2 + k^2 - 2jk + k}$ achieves its maximum at $j = 2$ and $j = k - 2$, when it has the value $2^{k^2 + 8 - 3k}$. Since $\binom{k}{j}$ is less than $2^k$, the number of graphs on $k$ vertices that have a split is less than $k * 2^{k^2 + 8 - 2k}$, which is a vanishingly small percentage of the $2^{k(k-1)}$ graphs on $k$ vertices for large values of $k$. We note that for large values of $k$, almost every graph on $k$ vertices is strongly connected [Pa], so restricting attention to strongly connected graphs does not alter this result.

**Conclusions.** This paper presents an $O(n^2)$ algorithm to determine whether a graph is prime with respect to the split decomposition. In a forthcoming paper [MS], we will show that this technique can be used to compute the entire undirected split decomposition in $O(n^2)$ time. We conjecture that an $O(n^2)$ algorithm for computing the entire directed split decomposition is also possible.

**Acknowledgments.** I would like to thank Ken Supowit and Bill Cunningham for several helpful discussions, and Mark Ellingham for finding a number of critical errors in an earlier draft of this paper. I would also like to thank the referees for a number of improvements to the organization of the paper.

## REFERENCES

[BM] H. Bandelt and H. M. Mulder, *Distance-hereditary graphs*, J. Combin. Theory Ser. B, 41 (1986), pp. 182–208.

[Bo1] A. Bouchet, *Digraph decompositions and Eulerian systems*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 323–337.

[Bo2] ———, *Reducing prime graphs and recognizing circle graphs*, Combinatorica, 7 (1987), pp. 243–254.

[CLS] D. G. Corneil, H. Lerchs, and L. Stewart Burlingham, *Complement reducible graphs*, Discrete Appl. Math., 3 (1981), pp. 163–174.

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

[GHS] C. P. Gabor, W. L. Hsu, and K. J. Supowit, *Recognizing circle graphs in polynomial time*, J. ACM, to appear.

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

[HM] P. L. Hammer and F. Maffray, *Completely separable graphs*, RUTCOR Research Report RRR 29-86, 1986.

[Hs] W.-L. Hsu, *$O(mn)$ Isomorphism algorithms for circular-arc graphs and circle graphs*, submitted for publication.

[Mo1] R. H. Möhring, *On the distribution of locally undecomposable relations and independent systems*, Methods Oper. Res., 42 (1981), pp. 33–48.

[Mo2] ———, *Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and Boolean functions*, Annals of Operations Research, 4 (1985), pp. 195–225.

[MR] R. H. Möhring and F. J. Radermacher, *Substitution decomposition for discrete structures and connections with combinatorial optimization*, Ann. Discrete Math., 19 (1984), pp. 257–356.

[MS] T.-H. Ma and J. Spinrad, *An $O(n^2)$ algorithm for the undirected split decomposition*, in preparation.

[Pa] I. Palasti, *On the strong connectedness of directed random graphs*, Studia Sci. Math. Hungarica, 1 (1966), pp. 205–214.

[Sp] J. Spinrad, *On comparability and permutation graphs*, SIAM J. Comput., 14 (1985), pp. 658–670.