

AN ALGEBRAIC CHARACTERIZATION OF COMPLETELY REGULAR CODES IN DISTANCE-REGULAR GRAPHS*

M. A. FIOŁ† AND E. GARRIGA†

Abstract. Given a vertex subset C of a distance-regular graph Γ on n vertices, it is shown that C is a completely regular code if and only if the number of vertices at maximum distance from C satisfies an expression in terms of the spectrum of Γ and some mean numbers computed from the distances among the vertices of C (the so-called “inner distribution” of C). For such codes, this result can be seen as an improvement of Delsarte’s linear programming method, since it gives stronger necessary conditions for their existence. As an application, a purely spectral characterization of those distance-regular graphs which are “edge-distance-regular” (that is, with every edge being a completely regular code with the same parameters) is derived.

Key words. distance-regular graph, local spectrum, orthogonal polynomials, completely regular code, edge-distance-regularity

AMS subject classifications. 05C50, 05C38

PII. S0895480100376496

1. Preliminaries. In this paper we always suppose that $\Gamma = (V, E)$ is a (simple, finite and connected) regular graph with vertex set $V = \{u, v, w, \dots\}$ and edge set $E = \{uv, wz, \dots\}$. Adjacency between vertices u and v is denoted by $u \sim v$. The *eccentricity* of a vertex u is $\text{ecc}(u) := \max_{v \in V} \text{dist}(u, v)$ and the *diameter* of the graph is $D := \max_{u \in V} \text{ecc}(u)$. Assuming that $\text{ecc}(u) = \varepsilon$, the set $\Gamma_k(u)$, $0 \leq k \leq \varepsilon$, represents the set of vertices at distance k from vertex u . Thus, the degree of vertex u is just the cardinality of $\Gamma_1(u)$ also written, for short, as $\Gamma(u)$. Similarly, the set of vertices at distance at most k from u is denoted by $N_k(u) := \Gamma_0(u) \cup \dots \cup \Gamma_k(u)$. The cardinalities of the above two sets are represented by $n_k(u) := |\Gamma_k(u)|$ and $s_k(u) := |N_k(u)|$. Analogous concepts can be defined for any vertex subset $C \subset V$ and, with self-explanatory notation, they are represented by $\Gamma_k(C)$ and $N_k(C)$. The characteristic vector of C will be denoted by $\rho C := \sum_{u \in C} e_u$, where e_u is the u th canonical vector. Consider the *adjacency matrix* \mathbf{A} of Γ , with spectrum

$$\text{sp } \Gamma := \text{sp } \mathbf{A} = \{\lambda_0^{m(\lambda_0)}, \lambda_1^{m(\lambda_1)}, \dots, \lambda_d^{m(\lambda_d)}\},$$

where the eigenvalues λ_i , $0 \leq i \leq d$, are in decreasing order, $\lambda_0 > \lambda_1 > \dots > \lambda_d$, and the superscripts denote multiplicities. The set of such eigenvalues is denoted by $\text{ev } \Gamma$, and their corresponding eigenspaces are

$$\mathcal{E}_i := \text{Ker}(\mathbf{A} - \lambda_i \mathbf{I}) \quad (0 \leq i \leq d).$$

The orthogonal projection onto the eigenspace \mathcal{E}_i is given by the polynomial

$$(1) \quad Z_i := \frac{1}{\phi_i} \prod_{j \neq i} (x - \lambda_j) \quad (0 \leq i \leq d),$$

*Received by the editors August 2, 2000; accepted for publication (in revised form) August 20, 2001; published electronically October 31, 2001. This work was supported in part by the Spanish Research Council (Comisi3n Interministerial de Ciencia y Tecnolog3a, CICYT) under project TIC 97-0963.

<http://www.siam.org/journals/sidma/15-1/37649.html>

†Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Jordi Girona, 1-3, M3dul C3, Campus Nord, 08034 Barcelona, Spain (fiol@mat.upc.es, egarriga@mat.upc.es).

where $\phi_i := \prod_{j=0(j \neq i)}^d (\lambda_i - \lambda_j)$. The corresponding matrices $\mathbf{E}_i := Z_i(\mathbf{A})$ are called the (*principal*) *idempotents* of \mathbf{A} . For instance, in a (regular) graph on n vertices, $\mathbf{E}_0 = \frac{1}{n}\mathbf{J}$ where \mathbf{J} is the all-1 matrix. Accordingly, the idempotents satisfy $\mathbf{E}_i\mathbf{E}_j = \delta_{ij}\mathbf{E}_i$, $\mathbf{A}\mathbf{E}_i = \lambda_i\mathbf{E}_i$, and $\text{sp } \mathbf{E}_i = \{1^{m(\lambda_i)}, 0^{n-m(\lambda_i)}\}$. Moreover, since every polynomial $p \in \mathbb{R}_d[x]$ can be written in terms of the above interpolating polynomials as $p(x) = \sum_{i=0}^d p(\lambda_i)Z_i(x)$, we get

$$(2) \quad p(\mathbf{A}) = \sum_{i=0}^d p(\lambda_i)\mathbf{E}_i.$$

(In fact, the above holds for any rational function p defined at each eigenvalue of \mathbf{A} ; see Godsil [18].)

The (u, v) -entry of the idempotent \mathbf{E}_i is called the *crossed uv -multiplicity* of λ_i , and we denote it by $m_{uv}(\lambda_i)$ (see [16]). For instance, in our case of regularity, $m_{uv}(\lambda_0) = 1/n$ for any $u, v \in V$. Notice that from (2) we have $p(\mathbf{A})_{uv} = \sum_{i=0}^d m_{uv}(\lambda_i)p(\lambda_i)$. An interesting case occurs when we consider the diagonal entries

$$m_{uu}(\lambda_i) = (\mathbf{E}_i)_{uu} = \|\mathbf{E}_i\mathbf{e}_u\|^2 \geq 0,$$

denoted also by $m_u(\lambda_i)$, which are referred to as the (*local*) *u -multiplicities* of λ_i . In [15] it was noted that when the graph is “seen” from vertex u , the u -multiplicities play a similar role as the standard multiplicities, thus justifying the name.

Distance-regular graphs and distance polynomials. Recall that a graph Γ with diameter D is distance-regular whenever, for any two vertices $u, v \in V$ at distance $\text{dist}(u, v) = k$, $0 \leq k \leq D$, the *intersection numbers* $c_k := |\Gamma_{k-1}(u) \cap \Gamma(v)|$, $a_k := |\Gamma_k(u) \cap \Gamma(v)|$, and $b_k := |\Gamma_{k+1}(u) \cap \Gamma(v)|$ do not depend on the chosen vertices u and v but only on their distance k .

For every integer k ($0 \leq k \leq D$), the *distance- k matrix* \mathbf{A}_k of a graph Γ is defined as the 01-matrix with coefficients $(\mathbf{A}_k)_{uv} := 1$ if and only if $\text{dist}(u, v) = k$, and it corresponds to the adjacency matrix of the so-called *distance- k graph* Γ_k . As it is well known (see, e.g., [23]), the distance-regularity of the graph Γ is characterized by the existence of polynomials p_k of degree exactly k such that

$$\mathbf{A}_k = p_k(\mathbf{A}) \quad (0 \leq k \leq D).$$

These are called the *distance polynomials*, and one of their main characteristics is that they constitute an orthogonal system with respect to the inner product

$$(3) \quad \langle p, q \rangle_\Gamma := \frac{1}{n} \text{tr}(p(\mathbf{A})q(\mathbf{A})) = \sum_{i=0}^d \frac{m(\lambda_i)}{n} p(\lambda_i)q(\lambda_i) \quad (p, q \in \mathbb{R}_D[x])$$

(with “normalized weight function” $g_i := \frac{1}{n}m(\lambda_i)$, $0 \leq i \leq d$, since $\sum_{i=0}^d g_i = 1$). Indeed, from the definition of the distance matrices note that it must be

$$(4) \quad \langle p_k, p_l \rangle_\Gamma = \frac{1}{n} \text{tr}(\mathbf{A}_k\mathbf{A}_l) = \delta_{kl}n_k,$$

where $n_k = |\Gamma_k(u)| = \|p_k\|_\Gamma^2 = p_k(\lambda_0)$ for any $u \in V$. Such an orthogonality relation greatly facilitates the computation of the different parameters of Γ . (For a survey about this, see [10].) Thus, the *intersection numbers* p_{ij}^k , $0 \leq i, j, k \leq d$, of the graph,

defined by $p_{ij}^k := |\Gamma_i(u) \cap \Gamma_j(v)|$, where $\text{dist}(u, v) = k$ (which generalize the above a_k , b_k , and c_k), are just the Fourier coefficients of $p_i p_j$ in terms of the basis $\{p_k\}_{0 \leq k \leq d}$:

$$(5) \quad p_{ij}^k = \frac{\langle p_i p_j, p_k \rangle_\Gamma}{\|p_k\|_\Gamma^2} = \frac{1}{p_k(\lambda_0)} \sum_{h=0}^d \frac{m(\lambda_h)}{n} p_i(\lambda_h) p_j(\lambda_h) p_k(\lambda_h) \quad (0 \leq i, j, k \leq d).$$

Also, the eigenvalues' multiplicities can be computed by using any of the following expressions:

$$(6) \quad m(\lambda_i) = \frac{\phi_0 p_d(\lambda_0)}{\phi_i p_d(\lambda_i)} = n \left(\sum_{j=0}^d \frac{p_j(\lambda_i)^2}{p_j(\lambda_0)} \right)^{-1} \quad (0 \leq i \leq d),$$

where $\phi_i = \prod_{j=0(j \neq i)}^d (\lambda_i - \lambda_j)$ (see, e.g., Bannai and Ito [1] and Biggs [2]). The value at λ_0 of the highest degree polynomial p_d can be computed, in turn, from the spectrum using the formula

$$(7) \quad p_d(\lambda_0) = n \left(\sum_{i=0}^d \frac{\pi_0^2}{m(\lambda_i) \pi_i^2} \right)^{-1},$$

where the π_i 's are moment-like parameters defined by $\pi_i := |\phi_i| = \prod_{j=0(j \neq i)}^d |\lambda_i - \lambda_j|$, $0 \leq i \leq d$ (see [11]).

In this work, we also use the fact that the distance polynomials allow us to express the idempotents \mathbf{E}_l , $0 \leq l \leq d$, of a distance-regular graph in a way closely related to its structure (see also [9]). Indeed, if we write the polynomials in (1) as $Z_l = \sum_{k=0}^d m_{kl} p_k$, then the constants m_{kl} are no more than the Fourier coefficients of Z_l in terms of the basis $\{p_k\}_{0 \leq k \leq d}$:

$$m_{kl} = \frac{\langle Z_l, p_k \rangle}{\|p_k\|^2} = \frac{1}{p_k(\lambda_0)} \sum_{i=0}^d g_i Z_l(\lambda_i) p_k(\lambda_i) = g_l \frac{p_k(\lambda_l)}{p_k(\lambda_0)},$$

which gives

$$(8) \quad \mathbf{E}_l = g_l \sum_{k=0}^d \frac{p_k(\lambda_l)}{p_k(\lambda_0)} p_k(\mathbf{A}) = \frac{m(\lambda_l)}{n} \sum_{k=0}^d \frac{p_k(\lambda_l)}{p_k(\lambda_0)} \mathbf{A}_k \quad (0 \leq l \leq d).$$

Thus, considering the corresponding (u, v) -entries of the above matrices, we have the following useful result.

LEMMA 1.1. *Let Γ be a distance-regular graph with $\text{sp } \Gamma := \{\lambda_0^{m(\lambda_0)}, \lambda_1^{m(\lambda_1)}, \dots, \lambda_d^{m(\lambda_d)}\}$. Then, for any two vertices $u, v \in V(\Gamma)$ at distance $\text{dist}(u, v) = k$, the crossed uv -multiplicities of λ_l , $0 \leq l \leq d$, depend only on k and l and are given by*

$$m_{uv}(\lambda_l) = m_{kl} = \frac{m(\lambda_l)}{n} \frac{p_k(\lambda_l)}{p_k(\lambda_0)} \quad (0 \leq l \leq d).$$

The fact that the crossed uv -multiplicities depend only on $\text{dist}(u, v)$ has been used by various authors studying distance-regular graphs and their generalizations. (See, e.g., Godsil [17, 18], who also gave an explicit formula for computing these entries.)

The local spectrum of a vertex subset. The concept of local multiplicities can be generalized to any vertex subset C . Indeed, if we consider its normalized characteristic vector

$$\mathbf{e}_C := \frac{1}{\sqrt{|C|}} \boldsymbol{\rho} C = \frac{1}{\sqrt{|C|}} \sum_{u \in C} \mathbf{e}_u,$$

the C -multiplicity of the eigenvalue λ_i is just

$$\begin{aligned} m_C(\lambda_i) &:= \|\mathbf{E}_i \mathbf{e}_C\|^2 = \langle \mathbf{E}_i \mathbf{e}_C, \mathbf{e}_C \rangle = \frac{1}{|C|} \sum_{u,v \in C} \langle \mathbf{E}_i \mathbf{e}_u, \mathbf{e}_v \rangle \\ (9) \quad &= \frac{1}{|C|} \sum_{u,v \in C} m_{uv}(\lambda_i) \quad (0 \leq i \leq d). \end{aligned}$$

Notice that the sequence of C -multiplicities $(m_C(\lambda_0), m_C(\lambda_1), \dots, m_C(\lambda_d))$ corresponds in fact to the so-called *MacWilliams transform* of the vector \mathbf{e}_C (see, e.g., [7]). Also note that, since \mathbf{e}_C is a unit vector, we have $\sum_{i=0}^d m_C(\lambda_i) = 1$. Moreover, we see that, for $C \neq \emptyset$, the C -multiplicity of λ_0 is $m_C(\lambda_0) = |C|/|V| > 0$. In particular, when C is a single vertex u , the $\{u\}$ -multiplicities of λ_i correspond to the above u -multiplicities.

These parameters are now relevant when studying the graph from vertex subset C . Before giving an example, notice that, for any polynomial p ,

$$\begin{aligned} \langle p(\mathbf{A}) \mathbf{e}_C, \mathbf{e}_C \rangle &= \left\langle \sum_{i=0}^d p(\lambda_i) \mathbf{E}_i \mathbf{e}_C, \mathbf{e}_C \right\rangle = \sum_{i=0}^d p(\lambda_i) \langle \mathbf{E}_i \mathbf{e}_C, \mathbf{e}_C \rangle \\ (10) \quad &= \sum_{i=0}^d m_C(\lambda_i) p(\lambda_i), \end{aligned}$$

where we have used (2). As mentioned above, these expressions allow us to know something about the structure of the graph when seen from C . For instance, using (10) with $p = x^l$ we have that the number of walks of length l from (the vertices of) C to itself is given by

$$a_{CC}^{(l)} := \sum_{u,v \in C} (\mathbf{A}^l)_{uv} = |C| \langle \mathbf{A}^l \mathbf{e}_C, \mathbf{e}_C \rangle = |C| \sum_{i=0}^d m_C(\lambda_i) \lambda_i^l \quad (l \geq 0).$$

If $\mu_0 (= \lambda_0) > \mu_1 > \dots > \mu_{d_C}$ represent the eigenvalues in $\text{ev } \Gamma$ with nonzero C -multiplicities, then the (*local*) C -spectrum of C is

$$\text{sp } C := \{\mu_0^{m_C(\mu_0)}, \mu_1^{m_C(\mu_1)}, \dots, \mu_{d_C}^{m_C(\mu_{d_C})}\},$$

where $d_C (\leq d)$ is called the *dual degree* of C . Also, the *strength* of C is the minimum integer $t \geq 0$ such that $m_C(\lambda_{t+1}) \neq 0$. (In other words, $\lambda_{t+1} = \mu_1$.) Now consider the polynomial $H_C \in \mathbb{R}_{d_C}[x]$ defined by $H_C(\lambda_0) = n/|C|$ and $H_C(\mu_i) = 0$ for any $1 \leq i \leq d_C$. Then again using (2), and the expressions for H_C and \mathbf{E}_0 , we have

$$(11) \quad H_C(\mathbf{A}) \mathbf{e}_C = \sum_{i=0}^{d_C} H_C(\mu_i) \mathbf{E}_i \mathbf{e}_C = H_C(\lambda_0) \mathbf{E}_0 \mathbf{e}_C = \frac{1}{|C|} \mathbf{J} \mathbf{e}_C = \frac{1}{\sqrt{|C|}} \mathbf{j}.$$

This polynomial is unique and is called the *C-Hoffman polynomial* because of the analogy with the well-known Hoffman polynomial H satisfying $H(\mathbf{A}) = \mathbf{J}$. By considering the v th components of both vectors in (11), $v \in V$, we obtain that $\text{dist}(v, C) \leq \text{dgr } H_C = d_C$. Hence, the eccentricity of a vertex set C with $d_C + 1$ distinct eigenvalues satisfies

$$(12) \quad \text{ecc}(C) \leq d_C.$$

(This is a well-known result in coding theory, where $\text{ecc}(C) := \max_{v \in V} \text{dist}(C, v)$ corresponds to the ‘‘covering radius’’ of code C ; see, e.g., [18].)

The predistance polynomials. Given a graph $\Gamma = (V, E)$, let us consider a (nonempty) vertex subset $C \subset V$. Let \mathcal{M}_C denote the mesh of the C -eigenvalues μ_i with respective local multiplicities $m_C(\mu_i)$, $0 \leq i \leq d_C$. Now consider the following scalar product in $\mathbb{R}[x]/(\phi)$, where (ϕ) is the ideal of $\mathbb{R}[x]$ generated by the polynomial $\phi := \prod_{i=0}^{d_C} (x - \mu_i)$,

$$(13) \quad \langle f, g \rangle_C := \sum_{i=0}^{d_C} m_C(\mu_i) f(\mu_i) g(\mu_i)$$

with weight function $g_i := m_C(\mu_i)$, which, from the definition of the local multiplicities, is again normalized as follows: $\sum_{i=0}^{d_C} g_i = 1$. Then we define the *predistance polynomials* p_k^C , $0 \leq k \leq d_C$, as the sequence of orthogonal polynomials with respect to (13), with $\text{dgr } p_k = k$, *normalized* in such a way that

$$(14) \quad \|p_k^C\|_C^2 := \langle p_k^C, p_k^C \rangle_C = p_k^C(\mu_0).$$

Thus, $p_0^C = 1$. Like any orthogonal sequence, these polynomials satisfy a three-term recurrence of the form

$$(15) \quad \begin{aligned} xp_0^C &= a_0 p_0^C + c_1 p_1^C, \\ xp_k^C &= b_{k-1} p_{k-1}^C + a_k p_k^C + c_{k+1} p_{k+1}^C \quad (1 \leq k \leq d_C - 1), \\ xp_{d_C}^C &= b_{d_C-1} p_{d_C-1}^C + a_{d_C} p_{d_C}^C. \end{aligned}$$

Moreover, it can be shown that its normalization condition (14) is equivalent to any of the following requirements (see [14]):

$$a_k + b_k + c_k = \mu_0 \quad (0 \leq k \leq d_C) \quad \text{or} \quad q_{d_C}^C := \sum_{k=0}^{d_C} p_k^C = H_C$$

(where $c_0 = b_{d_C} = 0$). Let us now consider the sum polynomials $q_k^C := \sum_{h=0}^k p_h^C$, $0 \leq k \leq d_C$, which, like the predistance polynomials, satisfy $q_k^C(\lambda_0) = \|q_k^C\|_C^2$. The following result corresponds to the regular case of a theorem proved in [12].

THEOREM 1.2. *Let C be a vertex subset of a (regular) graph Γ with predistance polynomials p_k^C , $0 \leq k \leq d$. Then, for any polynomial $q \in \mathbb{R}_k[x]$,*

$$(16) \quad \frac{q(\lambda_0)^2}{\|q\|_C^2} \leq \frac{|N_k(C)|}{|C|},$$

and equality is attained if and only if

$$(17) \quad \frac{1}{\|q\|_C} q(\mathbf{A}) \mathbf{e}_C = \mathbf{e}_{N_k},$$

where e_{N_k} represents the normalized characteristic vector of $N_k(C)$. Moreover, if this is the case, q is any multiple of q_k^C , whence (16) and (17) become

$$(18) \quad q_k^C(\lambda_0) = \frac{|N_k(C)|}{|C|} \quad \text{and} \quad q_k^C(\mathbf{A})\rho C = \rho N_k(C).$$

2. Completely regular codes in distance-regular graphs. This section is devoted to study the results on which our characterization of a completely regular code in a distance-regular graph Γ is based. The key point is that, in a distance-regular graph, the local multiplicities of a vertex subset C can easily be computed from the distance polynomials of Γ , its spectrum, and the inner distribution of C . As a by-product, the nonnegativity of such multiplicities leads to Delsarte-like bounds for the maximum possible number of subgraphs of a distance-regular graph isomorphic to a given subgraph. We begin by recalling some of the main concepts and results involved.

Distance-regularity around a set. Let $C \subset V$ be a vertex subset of a graph Γ with $\text{ecc}(C) = \varepsilon$. Then we say that Γ is *distance-regular around C* if the distance partition $V = C_0 \cup C_1 \cup \dots \cup C_\varepsilon$, where $C_k := \Gamma_k(C)$, is *regular*; that is, the numbers

$$c_k(u) := |\Gamma(u) \cap C_{k-1}|, \quad a_k(u) := |\Gamma(u) \cap C_k|, \quad b_k(u) := |\Gamma(u) \cap C_{k+1}|,$$

where $u \in C_k$, $0 \leq k \leq \varepsilon$, depend only on the value of k but not on the chosen vertex u . These numbers are called the (*local*) *C -intersection numbers* of Γ , and the array

$$\iota(C) := \begin{pmatrix} 0 & c_1 & \cdots & c_{\varepsilon-1} & c_\varepsilon \\ a_0 & a_1 & \cdots & a_{\varepsilon-1} & a_\varepsilon \\ b_0 & b_1 & \cdots & b_{\varepsilon-1} & 0 \end{pmatrix}$$

is referred to as the *C -intersection array* where, by convention, $c_0 = b_\varepsilon = 0$. Since each column of such a matrix sums to λ_0 (recall that Γ is regular), it is usual to write the intersection array in its simplified form

$$\iota(C) = \{b_0, b_1, \dots, b_{\varepsilon-1}; c_1, c_2, \dots, c_\varepsilon\}.$$

The set C is also referred to as a *completely regular set* or *completely regular code*. Since their introduction in Delsarte's thesis [6], these structures have deserved special attention by authors such as Courteau and Montpetit [5], Godsil [18], Martin [20], and Neumaier [22], among others. As in the case of distance-regularity, it is known that a graph $\Gamma = (V, E)$ is distance-regular around a set $C \subset V$, with eccentricity ε , if and only if there exist a sequence of polynomials $p_0, p_1, \dots, p_\varepsilon$, with $\text{dgr } p_k = k$, such that

$$\rho C_k = p_k(\mathbf{A})\rho C \quad (0 \leq k \leq \varepsilon)$$

(see [12]). If this is the case, such polynomials coincide with the predistance polynomials defined above, and they are just called the *distance polynomials*. Hence, they are orthogonal with respect to the scalar product (13) and satisfy a three-term recurrence like (15) with coefficients being the elements of the C -intersection array. As a consequence of Theorem 1.2, the authors [12] obtained the following characterization of distance-regularity around a set C in terms of its local spectrum.

THEOREM 2.1. *Let $\Gamma = (V, E)$ be a regular graph. A vertex subset $C \subset V$, with r vertices and local spectrum $\text{sp } C = \{\mu_0^{m_C(\mu_0)}, \mu_1^{m_C(\mu_1)}, \dots, \mu_{d_C}^{m_C(\mu_{d_C})}\}$, is a completely*

regular code if and only if the number of vertices at distance d_C from C , that is, $n_{d_C}(C) := |C_{d_C}|$, satisfies

$$(19) \quad \begin{aligned} n_{d_C}(C) &= r p_{d_C}^C(\mu_0) = r \left(\sum_{l=0}^{d_C} \frac{m_C(\mu_0)^2 \tilde{\pi}_0^2}{m_C(\mu_l) \tilde{\pi}_l^2} \right)^{-1} \\ &= \frac{n^2}{r} \left(\sum_{l=0}^{d_C} \frac{\tilde{\pi}_0^2}{m_C(\mu_l) \tilde{\pi}_l^2} \right)^{-1}, \end{aligned}$$

where $\tilde{\pi}_l := \prod_{h=0, h \neq l}^{d_C} |\mu_l - \mu_h|$, $0 \leq l \leq d_C$.

Local multiplicities of a vertex set. First note that, from (10), the number of ordered pairs (u, v) of vertices from C which are a distance k apart in a distance-regular graph Γ is given by

$$\sum_{u, v \in C} (\mathbf{A}_k)_{uv} = |C| \langle p_k(\mathbf{A}) \mathbf{e}_C, \mathbf{e}_C \rangle = |C| \sum_{i=0}^d m_C(\lambda_i) p_k(\lambda_i).$$

From this we see that the *mean number* of vertices v in C at distance k (in Γ) from a given vertex $u \in C$ is

$$(20) \quad r_k := \frac{1}{|C|} \sum_{u \in C} |\Gamma_k(u) \cap C| = \sum_{i=0}^d m_C(\lambda_i) p_k(\lambda_i) \quad (0 \leq k \leq d).$$

The numbers r_k , $0 \leq k \leq d$, are called the *inner distribution* of C and, as commented by Godsil [18], they determine the probability that a randomly chosen pair of vertices from C are at distance k . Notice that $r_0 = 1$ and $\sum_{k=0}^d r_k = |C|$ always. The following result shows how to use these numbers to compute the C -multiplicities of the graph.

PROPOSITION 2.2. *Let Γ be a distance-regular graph Γ with a given subset C of r vertices and inner distribution r_k , $0 \leq k \leq d$. Then the C -multiplicities are*

$$(21) \quad m_C(\lambda_l) = \frac{m(\lambda_l)}{n} \sum_{k=0}^d r_k \frac{p_k(\lambda_l)}{p_k(\lambda_0)} \quad (0 \leq l \leq d).$$

Proof. Let \mathbf{P} be the square $(d+1)$ -matrix with entries $(\mathbf{P})_{kl} = p_k(\lambda_l)$ which, because of the orthogonality relation (4) with respect to the scalar product (3), has inverse \mathbf{P}^{-1} with entries

$$(\mathbf{P}^{-1})_{lk} = \frac{1}{n} m(\lambda_l) \frac{p_k(\lambda_l)}{\|p_k\|^2} = \frac{1}{n} m(\lambda_l) \frac{p_k(\lambda_l)}{p_k(\lambda_0)}.$$

Moreover, using the (column) vectors $\mathbf{m}_C := (m_C(\lambda_0), m_C(\lambda_1), \dots, m_C(\lambda_d))^\top$ and $\mathbf{r} := (r_0, r_1, \dots, r_d)^\top$, the system of equations in (20) is $\mathbf{r} = \mathbf{P} \mathbf{m}_C$. Thus, $\mathbf{m}_C = \mathbf{P}^{-1} \mathbf{r}$ and yields (21). \square

Notice that Proposition 2.2 is essentially equivalent to Delsarte's identity $b = aQ$, which gives rise to the celebrated linear programming bound (see [7]). Let us now consider some interesting consequences of this result:

- The C -multiplicity of λ_0 is $m_C(\lambda_0) = \frac{m(\lambda_0)}{n} \sum_{k=0}^d r_k = \frac{r}{n}$, as we already knew.

- When C consists of a single vertex, say $C = \{u\}$, then $r_0 = r = 1$ and we get

$$m_C(\lambda_l) = \frac{m(\lambda_l) p_0(\lambda_l)}{n p_0(\lambda_0)} = \frac{m(\lambda_l)}{n} \quad (0 \leq l \leq d),$$

as it is also known, since any distance-regular graph is also *spectrally regular*; that is, $\text{sp}\{u\} = \text{sp}\{v\}$ for any $u, v \in V$; see [11]. (Spectrally regular graphs were first studied by Godsil and McKay [19] under the name of *walk-regular* graphs.)

- When C is a k -clique, that is, all its vertices are at distance k from each other, we have $r_0 = 1$, $r_k = r - 1$, and thus

$$m_C(\lambda_l) = \frac{m(\lambda_l)}{n} \left(1 + (r - 1) \frac{p_k(\lambda_l)}{p_k(\lambda_0)} \right) \quad (0 \leq l \leq d).$$

In particular, since $\min_{0 \leq l \leq d} p_k(\lambda_l) < 0$ for any $k \geq 1$ (as $p_k(\mathbf{A})$ is the adjacency matrix of Γ_k), the conditions $m_C(\lambda_l) \geq 0$, $1 \leq l \leq d$, yield the following upper bound for the maximum number ω_k of vertices mutually at distance k (*clique number* of Γ_k):

$$(22) \quad \omega_k = r \leq 1 - \frac{p_k(\lambda_0)}{\min_l p_k(\lambda_l)}.$$

- **Delsarte cliques.** If in (22) we take $k = 1$, then $p_k = x$ and we obtain that the clique number $\omega_1 = \omega$ of $\Gamma_1 = \Gamma$ satisfies the bound

$$\omega \leq 1 - \frac{\lambda_0}{\lambda_d}.$$

(This bound, in fact, holds for any regular graph; see, e.g., Godsil [18].) A clique which attains such a bound is called a *Delsarte clique*. Then the local multiplicities of the vertex set C of a Delsarte clique are

$$m_C(\lambda_i) = \frac{m(\lambda_i)}{n} \left(1 + (r - 1) \frac{\lambda_i}{\lambda_0} \right) = \frac{m(\lambda_i)}{n} \left(1 - \frac{\lambda_i}{\lambda_d} \right) \quad (0 \leq i \leq d).$$

Thus, $m_C(\lambda_i) = 0$ if and only if $i = d$, and hence, since $\text{dist}(C, v) \geq d - 1$ for some $v \in V$,

$$d - 1 \leq \text{ecc}(C) \leq d_C = d - 1,$$

so that the eccentricity of a Delsarte clique is exactly $d - 1$. Delsarte's cliques are completely regular codes. What is more, a clique which is a completely regular code is a Delsarte clique if and only if its eccentricity is $d - 1$ (see Godsil [18]).

- **Delsarte-like bounds.** In fact, the above reasoning, based on the nonnegativity of the local multiplicities, can be seen as a particular case of Delsarte's linear programming method to bound the size of a code C with prescribed parameters (ε, r, δ) . (As above, $\varepsilon = \text{ecc}(C)$, $r = |C|$, and δ is the *minimum distance* of C ; that is, $\delta := \min\{\text{dist}(u, v) : u, v \in C\}$, which yields $r_1 = \dots = r_{\delta-1} = 0$; see, e.g., [4, 7].) The same ideas can also be used to derive bounds on the maximum number of disjoint copies of some subgraph of a distance-regular graph. Before giving an example, we recall that the generic linear programming problem to be solved is the following. Given $r_0 = 1$ and some other (nonnegative) values of the inner distribution, say r_k , $k \in K \subset \{1, 2, \dots, d\}$,

$$\begin{aligned} & \text{maximize } r := \sum_{k=0}^d r_k \\ & \text{subject to } m_C(\lambda_l) \geq 0, \quad l = 0, 1, \dots, d; \\ & \quad r_k \geq 0, \quad k \in \{1, 2, \dots, d\} \setminus K. \end{aligned}$$

The following illustrates how to use this method to study the structure of the possible (57, 2)-Moore graph $\Gamma = (V, E)$. We shall show that the maximum number of independent pentagons (respectively, Petersen graphs) in such a graph—assuming that such a “monster” exists—is 100 (respectively, 55). As it is well known, Γ should be a distance-regular graph with intersection numbers $\{b_0, b_1; c_1, c_2\} = \{57, 56; 1, 1\}$, order $n = 57^2 + 1 = 3250$, spectrum $\text{sp } \Gamma = \{57^1, 7^{1729}, -8^{1520}\}$, and distance polynomials $p_0 = 1, p_1 = x, p_2 = x^2 - 57$.

First assume that $C \subset V$ is constituted by (the vertices of) m independent pentagons (that is, without edges between them). Then $r_0 = 1, r_1 = 2$, and the solution of the above problem gives $\max r_2 = 497$, which implies $m \leq r/5 = 100$, as claimed. Moreover, for the “extremal” inner distribution $\mathbf{r} = (1, 2, 497)$ we get $\mathbf{m}_C = (\frac{2}{13}, 0, \frac{11}{13})$, so that these 100 pentagons, if they exist, would dominate all the vertices of Γ , since $\text{ecc}(C) = 1$. Similarly, if C is set up by m copies of the Petersen graph, we have $r_0 = 1$ and $r_1 = 3$, yielding $\max r_2 = 546$ and $m \leq r/10 = 55$. Now, the extremal inner distribution $\mathbf{r} = (1, 3, 546)$ gives $\mathbf{m}_C = (\frac{11}{65}, 0, \frac{54}{65})$, whence such hypothetical 55 Petersen graphs would also dominate all vertices of Γ . It is fair to warn here that, as commented by Martin in [21], it is still an open problem to decide whether or not our Moore graph Γ should contain a single Petersen graph as an induced subgraph (a problem posed by Godsil).

3. Some quasi-spectral characterizations. From Theorem 2.1 and Proposition 2.2 we now have the following characterization of a completely regular code in a distance-regular graph.

THEOREM 3.1. *Let $\Gamma = (V, E)$ be a distance-regular graph on n vertices with $\text{sp } \Gamma = \{\lambda_0^{m(\lambda_0)}, \dots, \lambda_d^{m(\lambda_d)}\}$ and distance polynomials $\{p_k\}_{0 \leq k \leq d}$. A vertex subset $C \subset V$, with r vertices, inner distribution r_0, r_1, \dots, r_d , and local spectrum $\text{sp } C = \{\mu_0^{m_C(\mu_0)}, \dots, \mu_{d_C}^{m_C(\mu_{d_C})}\}$, is a completely regular code if and only if*

$$(23) \quad n_{d_C}(C) = \frac{n}{r} \left(\sum_{l=0}^{d_C} \frac{\tilde{\pi}_0^2}{\tilde{\pi}_l^2} \left(m(\mu_l) \sum_{k=0}^d \frac{p_k(\mu_l)}{p_k(\mu_0)} r_k \right)^{-1} \right)^{-1},$$

where $\tilde{\pi}_l := \prod_{h=0(h \neq l)}^{d_C} |\mu_l - \mu_h|, 0 \leq l \leq d_C$.

When C is a set with maximum possible eccentricity in Γ , that is, $\text{ecc}(C) = d$, we have $d_C = d, \tilde{\pi}_l = \pi_l$, and hence we get the following corollary.

COROLLARY 3.2. *Let $\Gamma = (V, E)$ be a distance-regular graph with n vertices, spectrum $\text{sp } \Gamma = \{\lambda_0, \lambda_1^{m(\lambda_1)}, \dots, \lambda_d^{m(\lambda_d)}\}$, and distance polynomials p_0, p_1, \dots, p_d . Then, a vertex subset $C \subset V$ with r vertices, $\text{ecc}(C) = d$, and inner distribution r_0, r_1, \dots, r_d is a completely regular code if and only if the number of vertices at distance d from C is*

$$(24) \quad n_d(C) = \frac{n}{r} \left(\sum_{l=0}^d \frac{\pi_0^2}{\pi_l^2} \left(m(\lambda_l) \sum_{k=0}^d \frac{p_k(\lambda_l)}{p_k(\lambda_0)} r_k \right)^{-1} \right)^{-1}.$$

Next, we give some examples of application of the above characterization results.

Codes in the odd graphs. First recall that, for a given integer $k \geq 2$, the *odd graph* O_k has as vertices the $(k-1)$ -subsets of a $(2k-1)$ -set, and adjacency is defined by void intersection. (For a detailed description of these distance-regular graphs, see, e.g., [2, 3]). In particular, we shall consider here some vertex subsets of the odd graph O_4 , which is a 4-regular graph on $n = 35$ vertices, with diameter $D = 3$, spectrum $\text{sp } O_4 = \{4^1, 2^{14}, -1^{14}, -3^6\}$, and distance polynomials $p_0 = 1$, $p_1 = x$, $p_2 = x^2 - 4$, $p_3 = \frac{1}{2}(x^3 - 7x)$.

- $C = uv \in E$. (That is, C is constituted by the vertices of an edge.) $\mathbf{r} = (1, 1, 0, 0)$, $\mathbf{m}_C = (\frac{2}{35}, \frac{3}{5}, \frac{3}{10}, \frac{3}{70})$, and (24) gives $n_3(C) = 9$, which is the correct value. Thus, C is a completely regular code. (In fact, we will see later that this observation holds for any odd graph O_k .)
- $C = H$ (a hexagon). $\mathbf{r} = (1, 2, 2, 1)$. $\mathbf{m}_C = (\frac{6}{35}, \frac{11}{15}, \frac{1}{15}, \frac{1}{35})$, which gives $n_3(C) = 11/7$, thus violating the integrality condition, and C cannot be completely regular.

Codes in antipodal distance-regular graphs. Let us prove that any proper subset C of a fiber (that is, the set of vertices mutually at distance d) in an s -antipodal distance-regular graph is a completely regular code. A (quasi-) spectral characterization of these graphs can be found in [8], where it was shown that a regular graph Γ , with eigenvalues $\lambda_0 > \lambda_1 > \dots > \lambda_d$, is an s -antipodal distance-regular graph if and only if the distance graph Γ_d is constituted by $\frac{1}{2} \sum_{i=0}^d \frac{\pi_0}{\pi_i}$ disjoint copies of the complete graph K_s .

First, we particularize the above corollary for the case when all vertices are at distance d from each other.

COROLLARY 3.3. *Let Γ be a distance-regular graph as above. A vertex subset $C \subset V$, with r vertices, $\text{ecc}(C) = d$, and inner distribution $r_0 = 1$, $r_d = r - 1$, is a completely regular code if and only if*

$$(25) \quad n_d(C) = \frac{n}{r} \left(\sum_{l=0}^d \frac{\phi_0}{\phi_l} \left(\frac{p_d(\lambda_0)}{p_d(\lambda_l)} + r - 1 \right)^{-1} \right)^{-1}.$$

Proof. Using (6), the expression in the inner parentheses of (23) becomes

$$m(\lambda_l) \sum_{k=0}^d \frac{p_k(\lambda_l)}{p_k(\lambda_0)} r_k = \frac{\phi_0 p_d(\lambda_0)}{\phi_l p_d(\lambda_l)} \left(1 + (r-1) \frac{p_d(\lambda_l)}{p_d(\lambda_0)} \right) = \frac{\phi_0}{\phi_l} \left(\frac{p_d(\lambda_0)}{p_d(\lambda_l)} + r - 1 \right),$$

whence the result follows. \square

Now, if Γ is s -antipodal, $p_d(\mathbf{A})$ corresponds to the adjacency matrix of n/s copies of the complete graph K_s , whence $p_d(\lambda_i) = s - 1$ for every even i , and $p_d(\lambda_i) = -1$ for any odd i . Then, for a vertex subset C as in the above corollary and $1 \leq r < s$, (25) gives

$$\frac{n}{r} \left(\frac{1}{r} \sum_{l \text{ even}} \frac{\pi_0}{\pi_l} + \frac{1}{s-r} \sum_{l \text{ odd}} \frac{\pi_0}{\pi_l} \right)^{-1} = s - r = n_d(C),$$

where we have used that, in any s -antipodal distance-regular graph,

$$\sum_{l \text{ even}} \frac{\pi_0}{\pi_l} = \sum_{l \text{ odd}} \frac{\pi_0}{\pi_l} = \frac{n}{s};$$

see [8, 16]. Consequently, by Corollary 3.3, the proper subset C of a fiber in Γ is a completely regular code.

Edge-distance-regularity. Inspired by the definition of a distance-regular graph, we say that a graph $\Gamma = (V, E)$ is *edge-distance-regular* when Γ is distance-regular around each of its edges, $C = \{u, v | u \sim v\}$, and it has the same C -intersection numbers. Thus, in an edge-distance-regular graph, every edge $e = uv$ is a completely regular code with parameters not depending on e . As it is readily checked, examples of these graphs are all bipartite distance-regular graphs. Two other instances are the (ubiquitous) Petersen graph $P(= O_3)$ and the above-mentioned O_4 . In this last subsection we address the natural question of deciding which (nonbipartite) distance-regular graphs are also edge-distance-regular. In fact, from our previous results we can give a complete answer to this question by deriving a pure spectral characterization of edge-distance-regularity. The result can be seen as a consequence of Theorem 3.1, and it reads as follows.

THEOREM 3.4. *A nonbipartite distance-regular graph $\Gamma = (V, E)$ with spectrum $\text{sp } \Gamma = \{\lambda_0^{m(\lambda_0)}, \lambda_1^{m(\lambda_1)}, \dots, \lambda_d^{m(\lambda_d)}\}$ and distance polynomials $\{p_k\}_{0 \leq k \leq d}$ is edge-distance-regular if and only if*

$$(26) \quad \sum_{l,h=0}^d \left(\frac{2\lambda_l}{\lambda_0 + \lambda_h} - 1 \right) \frac{1}{\pi_l^2 \pi_h^2 m(\lambda_l) m(\lambda_h)} = 0.$$

Proof. Let $C = \{u, v\}$ be the vertices of some edge $e = uv$ in a distance-regular graph Γ . Then, by (21), the C -multiplicities are

$$(27) \quad m_C(\lambda_l) = \frac{m(\lambda_l)}{n} \left(1 + \frac{\lambda_l}{\lambda_0} \right) \quad (0 \leq l \leq d).$$

Thus, the C -spectrum does not depend on the chosen edge e and, if Γ is distance-regular around e , such an independence property also holds for the corresponding C -intersection numbers (as they are uniquely determined from $\text{sp } C$; see [12]). Consequently, Γ is edge-distance-regular if and only if Theorem 3.1 holds. Since Γ is not bipartite, $-\lambda_0 < \lambda_l \leq \lambda_0$ and, from (27), $m_C(\lambda_l) > 0$ for any $0 \leq l \leq d$. Thus, $d_C = d$. Moreover, since Γ is distance-regular, the number of vertices at maximum distance d from edge uv is just the intersection number p_{dd}^1 . Hence, using (5),

$$(28) \quad \begin{aligned} n_d(C) = p_{dd}^1 &= \frac{1}{n\lambda_0} \sum_{l=0}^d m(\lambda_l) p_d^2(\lambda_l) \lambda_l = \frac{p_d^2(\lambda_0)}{n\lambda_0} \sum_{l=0}^d \frac{\pi_0^2}{\pi_l^2} \frac{\lambda_l}{m(\lambda_l)} \\ &= \frac{n}{\lambda_0} \frac{\sum_{l=0}^d \frac{\pi_0^2}{\pi_l^2} \frac{\lambda_l}{m(\lambda_l)}}{\left(\sum_{l=0}^d \frac{\pi_0^2}{\pi_l^2} \frac{1}{m(\lambda_l)} \right)^2} = \frac{n}{\lambda_0 \pi_0^2} \frac{\sum_{l=0}^d \frac{\lambda_l}{\pi_l^2 m(\lambda_l)}}{\left(\sum_{l=0}^d \frac{1}{\pi_l^2 m(\lambda_l)} \right)^2}, \end{aligned}$$

where we have used (6) and (7).

Moreover, the right-hand expression in (24), with $r = r_0 + r_1 = 2$, becomes

$$(29) \quad \frac{n}{2} \left(\sum_{l=0}^d \frac{\pi_0^2}{\pi_l^2} \left(m(\lambda_l) \left[1 + \frac{\lambda_l}{\lambda_0} \right] \right)^{-1} \right)^{-1} = \frac{n}{2\pi_0^2 \lambda_0} \left(\sum_{l=0}^d \frac{1}{\pi_l^2 (\lambda_0 + \lambda_l) m(\lambda_l)} \right)^{-1}.$$

Consequently, equating (28) and (29) we get

$$2 \sum_{l=0}^d \frac{\lambda_l}{\pi_l^2 m(\lambda_l)} \sum_{h=0}^d \frac{1}{\pi_h^2 (\lambda_0 + \lambda_h) m(\lambda_h)} = \left(\sum_{l=0}^d \frac{1}{\pi_l^2 m(\lambda_l)} \right)^2 = \sum_{l,h=0}^d \frac{1}{\pi_l^2 \pi_h^2 m(\lambda_l) m(\lambda_h)},$$

whence the result follows. \square

As an application of the above result, it can be shown that the odd graphs O_k satisfy (26). Therefore, they are all examples of edge-distance-regular graphs. (We already mentioned the case $k = 3$, which corresponds to the Petersen graph, and the case $k = 4$.) This gives a nontrivial, although too simple for applications, infinite family of completely regular codes in the odd graphs, answering in the affirmative a question posed by Martin in [21]. The same conclusion can be derived from the combinatorial counterpart of Theorem 3.4, which is clear from the results in [3] (see also [13]), stating that a nonbipartite distance-regular graph Γ with diameter d is edge-distance-regular if and only if $a_1 = a_2 = \dots = a_{d-1} = 0$).

Acknowledgments. The authors sincerely acknowledge the referees for helpful comments and suggestions that led to a significant improvement of the manuscript.

REFERENCES

- [1] E. BANNAI AND T. ITO, *Algebraic Combinatorics I: Association Schemes*, Benjamin/Cummings, Menlo Park, CA, 1984.
- [2] N. BIGGS, *Algebraic Graph Theory*, Cambridge University Press, Cambridge, UK, 1974.
- [3] A. E. BROUWER, A. M. COHEN, AND A. NEUMAIER, *Distance-Regular Graphs*, Springer-Verlag, Berlin, 1989.
- [4] A. E. BROUWER AND W. H. HAEMERS, *Association schemes*, in Handbook of Combinatorics, Vol. 1, 2, Elsevier, Amsterdam, 1995, pp. 747–771.
- [5] B. COURTEAU AND A. MONTPETIT, *Dual distances of completely regular codes*, Discrete Math., 89 (1991), pp. 7–15.
- [6] P. DELSARTE, *An algebraic approach to the association schemes of coding theory*, Philips Res. Rep. Suppl., 10 (1973).
- [7] P. DELSARTE AND V. I. LEVENSHTAIN, *Association schemes and coding theory*, IEEE Trans. Inform. Theory, 44 (1998), pp. 2477–2504.
- [8] M. A. FIOLE, *An eigenvalue characterization of antipodal distance-regular graphs*, Electron. J. Combin., 4 (1997).
- [9] M. A. FIOLE, *On pseudo-distance-regularity*, Linear Algebra Appl., 323 (2001), pp. 145–165.
- [10] M. A. FIOLE, *Algebraic characterizations of distance-regular graphs*, Discrete Math., to appear.
- [11] M. A. FIOLE AND E. GARRIGA, *From local adjacency polynomials to locally pseudo-distance-regular graphs*, J. Combin. Theory Ser. B, 71 (1997), pp. 162–183.
- [12] M. A. FIOLE AND E. GARRIGA, *On the algebraic theory of pseudo-distance-regularity around a set*, Linear Algebra Appl., 298 (1999), pp. 115–141.
- [13] M. A. FIOLE AND E. GARRIGA, *Edge-Distance-Regular Graphs*, Research Report, Universitat Politècnica de Catalunya, Barcelona, Spain, 1999.
- [14] M. A. FIOLE AND E. GARRIGA, *Some applications of orthogonal polynomials of a discrete variable to graphs and codes*, in Proceedings of Alhambra 2000 (Symposium on Orthogonal Polynomials), Granada, Spain, 2000.
- [15] M. A. FIOLE, E. GARRIGA, AND J. L. A. YEBRA, *Locally pseudo-distance-regular graphs*, J. Combin. Theory Ser. B, 68 (1996), pp. 179–205.
- [16] M. A. FIOLE, E. GARRIGA, AND J. L. A. YEBRA, *Boundary graphs: The limit case of a spectral property*, Discrete Math., 226 (2001), pp. 155–173.
- [17] C. D. GODSIL, *Bounding the diameter of distance regular graphs*, Combinatorica, 8 (1988), pp. 333–343.
- [18] C. D. GODSIL, *Algebraic Combinatorics*, Chapman and Hall, New York, 1993.
- [19] C. D. GODSIL AND B. D. MCKAY, *Feasibility conditions for the existence of walk-regular graphs*, Linear Algebra Appl., 30 (1980), pp. 51–61.

- [20] W. J. MARTIN, *Completely Regular Subsets*, Ph.D. thesis, University of Waterloo, Waterloo, ON, Canada, 1992.
- [21] W. J. MARTIN, *Some open problems in algebraic combinatorics*, available online from <http://www.uwinnipeg.ca/~martin/RESEARCH/open.html>.
- [22] A. NEUMAIER, *Completely regular codes*, *Discrete Math.*, 106/107 (1992), pp. 353–360.
- [23] P. ROWLINSON, *Linear algebra*, in *Graph Connections*, L. W. Beineke and R. J. Wilson, eds., Oxford Lecture Ser. Math. Appl. 5, Oxford University Press, New York, 1997, pp. 86–99.

DETERMINING WHEN THE ABSOLUTE STATE COMPLEXITY OF A HERMITIAN CODE ACHIEVES ITS DLP BOUND*

T. BLACKMORE[†] AND G. H. NORTON[‡]

Abstract. Let g be the genus of the Hermitian function field H/\mathbb{F}_{q^2} and let $C_{\mathcal{L}}(D, mQ_{\infty})$ be a typical Hermitian code of length n . In [*Des. Codes Cryptogr.*, to appear], we determined the dimension/length profile (DLP) lower bound on the state complexity of $C_{\mathcal{L}}(D, mQ_{\infty})$. Here we determine when this lower bound is tight and when it is not.

For $m \leq \frac{n-2}{2}$ or $m \geq \frac{n-2}{2} + 2g$, the DLP lower bounds reach Wolf’s upper bound on state complexity and thus are trivially tight. We begin by showing that for about half of the remaining values of m the DLP bounds cannot be tight. In these cases, we give a lower bound on the absolute state complexity of $C_{\mathcal{L}}(D, mQ_{\infty})$, which improves the DLP lower bound.

Next we give a “good” coordinate order for $C_{\mathcal{L}}(D, mQ_{\infty})$. With this good order, the state complexity of $C_{\mathcal{L}}(D, mQ_{\infty})$ achieves its DLP bound (whenever this is possible). This coordinate order also provides an upper bound on the absolute state complexity of $C_{\mathcal{L}}(D, mQ_{\infty})$ (for those values of m for which the DLP bounds cannot be tight). Our bounds on absolute state complexity do not meet for some of these values of m , and this leaves open the question whether our coordinate order is best possible in these cases.

A straightforward application of these results is that if $C_{\mathcal{L}}(D, mQ_{\infty})$ is self-dual, then its state complexity (with respect to the lexicographic coordinate order) achieves its DLP bound of $\frac{n}{2} - \frac{g^2}{4}$, and, in particular, so does its absolute state complexity.

Key words. Hermitian code, state complexity, dimension/length profile bound

AMS subject classifications. 94B27, 94B12, 14H45

PII. S0895480100376435

1. Introduction. Let C be a linear code of length n . Many soft-decision decoding algorithms for C (such as the Viterbi algorithm and lower complexity derivatives of it) take place along a minimal trellis for C . The complexity of trellis decoding algorithms can be measured by various trellis complexities. The most common one is the state complexity $s(C)$ of C , which varies with the coordinate order of C . Since the number of operations required for Viterbi decoding of C is proportional to $s(C)$, it is desirable that $s(C)$ be small. A classical upper bound for $s(C)$ is the Wolf bound $W(C) = \min\{\dim(C), n - \dim(C)\}$ [9]. It is well known that if C is a Reed–Solomon code, then $s(C) = W(C)$.

Let $[C]$ denote the set of codes equivalent to C by a change of coordinate order. We write $s[C]$ for the minimum of $s(C)$ over all coordinate orders of C and call it the *absolute state complexity* of C . (We note that state-complexity notation and terminology varies in the literature. For example, state complexity is called minimal trellis size in [2]; absolute state complexity is called absolute minimal trellis size in [2] and minimal state complexity in [13].) Finding a coordinate order of C that achieves $s[C]$ is called the “art of trellis decoding” in [10] since exhaustive computation of $s(C)$

*Received by the editors August 7, 2000; accepted for publication (in revised form) July 25, 2001; published electronically October 31, 2001. This research was supported by the UK Engineering and Physical Sciences Research Council under grant L88764 at the Algebraic Coding Research Group, Centre for Communications Research, University of Bristol.

<http://www.siam.org/journals/sidma/15-1/37643.html>

[†]Infinion Technologies, Stoke Gifford, BS34 8HP, UK (tim.blackmore@infinion.com).

[‡]Department of Mathematics, University of Queensland, Brisbane 4072, Australia (ghn@maths.uq.edu.au, <http://www.maths.uq.edu.au/~ghn>).

over all possible coordinate orders of C is infeasible, even for quite short codes. An important step towards attaining this goal is determining good lower bounds on $s[C]$.

The dimension/length profile (DLP) of C is a deep property which is equivalent to the generalized weight hierarchy (GWH) of C . (For a survey of GWH, see [15].) The DLP of C is independent of the coordinate order of C and provides a natural lower bound $\nabla(C)$ for $s[C]$. For example, if C is a Reed–Solomon code, then $\nabla(C) = W(C)$ [9], so that $s[C]$ is as bad as possible and uninteresting. However, determining when $\nabla(C) = s[C]$ is still important. An obvious and useful way of doing this is to find a coordinate order of C for which $s(C) = \nabla(C)$. In particular, this provides one route to the art of trellis decoding. It is also important to develop methods for determining when $\nabla(C) < s(C)$ and, in these cases, to improve on $\nabla(C)$.

Geometric Goppa codes generalize Reed–Solomon codes. Hermitian codes are widely studied geometric Goppa codes which are longer than Reed–Solomon codes and have very good parameters for their lengths. Let q be a fixed prime power, $n = q^3$, and $g = \binom{q}{2}$. For $m \in [0, n + 2g - 2]$, we write $C_{\mathcal{L}}(D, mQ_{\infty})$ for a typical Hermitian code of length n defined over \mathbb{F}_{q^2} . In [5], we determined $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$ using some of the GWH of Hermitian codes obtained in [11, 16]. (The complete GWH of Hermitian codes has subsequently appeared in [1].) From [5], we have $s(C_{\mathcal{L}}(D, mQ_{\infty})) = W(C_{\mathcal{L}}(D, mQ_{\infty}))$ for $m < \frac{n-1}{2}$ or $m > \frac{n-3}{2} + 2g$, so we restrict ourselves to the interesting Hermitian codes, i.e., to $C_{\mathcal{L}}(D, mQ_{\infty})$ with $m \in I(n, g) = [\frac{n-1}{2}, \frac{n-3}{2} + 2g]$.

Here we determine precisely when $\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = s(C_{\mathcal{L}}(D, mQ_{\infty}))$. In the process, we exhibit a good coordinate order which often gives $s(C_{\mathcal{L}}(D, mQ_{\infty})) < W(C_{\mathcal{L}}(D, mQ_{\infty}))$. We also improve on the DLP bound (when it is strictly less than the state complexity).

“Points of gain and fall” were introduced in [3, 4, 6, 7] to help determine the state complexity of certain generalizations of Reed–Muller codes. For these codes, the points of gain and fall had particularly nice characterizations. For Hermitian codes, however, their characterization is not quite as nice, and so our approach is slightly different. We describe a coordinate order giving $C_m \in [C_{\mathcal{L}}(D, mQ_{\infty})]$ and characterize the points of gain and fall of C_m . We also characterize these points of gain and fall in terms of runs. This has the advantage of greatly reducing (from n to $q + 1$) the number of trellis depths needed to find $s(C_m)$.

The paper is arranged as follows. Section 2 contains terminology, notation, and some previous results that will be used throughout the paper. The paper proper begins with section 3. Here we show that, for $m \in I(n, g)$, just under half of the Hermitian codes cannot attain their DLP bound. In these cases we give an improvement of the DLP bound, written $\nabla^i(C_{\mathcal{L}}(D, mQ_{\infty}))$.

The main goal of section 4 is to characterize the points of gain and fall of C_m in runs. In section 5 we determine $s(C_m)$ using section 4. We show that $s(C_m) = \nabla(C_m)$ for just over half the $m \in I(n, g)$. Thus we have determined precisely when the DLP bound for Hermitian codes is tight. Furthermore, $s(C_m) = \nabla^i(C_m)$ for around a further quarter (respectively, $1/q$) of $m \in I(n, g)$ when q is odd (respectively, even).

In conclusion, we have found $s[C_m]$ for three quarters (respectively, one half) of the $m \in I(n, g)$ when q is odd (respectively, even). For the remaining $m \in I(n, g)$, we do not know a better coordinate order (than that described in section 4) nor a better bound (than that given in section 3). Thus, although we have reduced the possible range of $s[C_m]$, some of its actual values remain open. Finally, our method of characterizing points of gain and fall is essentially the same as the one used to determine $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$ in [5] and may be able to be used quite generally in

determining DLP bounds and state complexity.

The state complexity of Hermitian codes has also been studied in [13]. For a stronger version of [13, Proposition 1] (an application of Clifford's theorem), see [5, Proposition 3.4]. Also, Example 5.11 below generalizes the main result of [13] to arbitrary self-dual Hermitian codes. An initial account of some of these results appeared in [8].

2. Terminology, notation, and background.

State complexity. Let C be a linear code of length n and $0 \leq i \leq n$. The state space dimension of C at depth i is

$$(1) \quad s_i(C) = \dim(C) - \dim(C_{i,-}) - \dim(C_{i,+}),$$

where $C_{i,-} = \{c \in C : c_{i+1} = \dots = c_n = 0\}$ and $C_{i,+} = \{c \in C : c_1 = \dots = c_i = 0\}$. The *state complexity of C* is $s(C) = \max\{s_i(C) : 0 \leq i \leq n\}$. It is well known that $s(C^\perp) = s(C)$. A simple upper bound on $s(C)$ (and hence on $s[C]$) is the Wolf bound $W(C) = \min\{\dim(C), n - \dim(C)\}$. We write $[C]$ for the set of codes equivalent to C by a change of coordinate order; i.e., $C' \in [C]$ if and only if there exists a permutation (l_1, \dots, l_n) of $(1, \dots, n)$ such that $C' = \{(c_{l_1}, \dots, c_{l_n}) : (c_1, \dots, c_n) \in C\}$. Then we define the *absolute state complexity of C* to be

$$s[C] = \min\{s(C') : C' \in [C]\}.$$

The DLP of C is $(k_0(C), \dots, k_n(C))$, where $k_i(C) = \max\{\dim(C_J) : |J| = i\}$. Clearly, $\dim(C_{i,-}) \leq k_i(C)$ and $\dim(C_{i,+}) \leq k_{n-i}(C)$, so that $s_i(C) \geq \dim(C) - k_i(C) - k_{n-i}(C)$. The *DLP bound on $s_i(C)$* is

$$\nabla_i(C) = \dim(C) - k_i(C) - k_{n-i}(C),$$

and the *DLP bound on $s(C)$* is $\nabla(C) = \max\{\nabla_i(C) : 0 \leq i \leq n\}$. We will use *DLP bound to mean $\nabla(C)$ for some C* . It is well known that $\nabla(C^\perp) = \nabla(C)$. Since $\nabla(C)$ is independent of the coordinate order of C , $\nabla(C) \leq s[C]$. If $s[C] = \nabla(C)$, we say that C is *DLP-tight*; e.g., if $\nabla(C) = W(C)$, then C is DLP-tight.

Hermitian codes. Our terminology and notation for Hermitian codes for the most part follow [14]. We write H/\mathbb{F}_{q^2} for the Hermitian function field. Thus $H = \mathbb{F}_{q^2}[x, y]$, where x is transcendental over \mathbb{F}_{q^2} and $y^q + y = x^{q+1}$ is the minimal polynomial of y over $\mathbb{F}_{q^2}[x]$. The genus of H/\mathbb{F}_{q^2} is $g = \binom{q}{2} > 0$. We write \mathbb{P}_H for the set of places of H/\mathbb{F}_{q^2} and \mathcal{D}_H for the divisor group of H/\mathbb{F}_{q^2} . For $Q \in \mathbb{P}_H$ and $z \in H/\mathbb{F}_{q^2}$, we write $v_Q(z)$ for the valuation of z at Q . Thus $v_Q(z) < 0$ if and only if Q is a pole of z and $v_Q(z) > 0$ if and only if Q is a zero of z . Also, $(z) \in \mathcal{D}_H$ is given by $(z) = \sum_{Q \in \mathbb{P}_H} v_Q(z)Q$ and for $A \in \mathcal{D}_H$, $\mathcal{L}(A) = \{z \in H/\mathbb{F}_{q^2} : (z) \geq -A\} \cup \{0\}$.

There are $q^3 + 1$ places of degree one in \mathbb{P}_H . One of these is the place at infinity, which we denote Q_∞ . We denote the others as Q_1, \dots, Q_{q^3} . For the rest of the paper, unless otherwise stated, $n = q^3$. We put $D = \sum_{j=1}^n Q_j$. For an integer m , $\mathcal{L}(mQ_\infty) = \{z \in H/\mathbb{F}_{q^2} : (z) \geq -mQ_\infty\} \cup \{0\}$. The Hermitian codes over \mathbb{F}_{q^2} are $C_{\mathcal{L}}(D, mQ_\infty) = \{z(Q_{l_1}), \dots, z(Q_{l_n}) : z \in \mathcal{L}(mQ_\infty)\}$ for some permutation (l_1, \dots, l_n) of $(1, \dots, n)$. Strictly speaking, the code $C(D, mQ_\infty)$ depends on the permutation (l_1, \dots, l_n) of $(1, \dots, n)$ and may be better denoted $C_{\mathcal{L}}(Q_{l_1}, \dots, Q_{l_n}; mQ_\infty)$. However, this notation is cumbersome and $C_{\mathcal{L}}(D, mQ_\infty)$ is standard. Unless otherwise stated, when we write $C_{\mathcal{L}}(D, mQ_\infty)$ we have some fixed but arbitrary coordinate order in mind.

From the usual expression for the dimension of geometric Goppa codes,

$$\dim(C_{\mathcal{L}}(D, mQ_{\infty})) = \dim(mQ_{\infty}) - \dim(mQ_{\infty} - D).$$

When m is understood, $k = \dim(C_{\mathcal{L}}(D, mQ_{\infty}))$ unless stated otherwise. The abundance of $C_{\mathcal{L}}(D, mQ_{\infty})$ is $\dim(mQ_{\infty} - D)$. For $m < n$, the abundance is 0 and the code is nonabundant. For $m < 0$, $C_{\mathcal{L}}(D, mQ_{\infty}) = \{0\}$ and for $m > n + 2g - 2$, $C_{\mathcal{L}}(D, mQ_{\infty}) = \mathbb{F}_q^n$, so we restrict our attention to $m \in [0, n + 2g - 2]$. With $m^{\perp} = n + 2g - 2 - m$, the dual of $C_{\mathcal{L}}(D, mQ_{\infty})$ is $C_{\mathcal{L}}(D, m^{\perp}Q_{\infty})$.

Let $\Pi : \mathbb{N} \rightarrow \mathbb{N} \cup \{0\}$ be the pole number sequence of Q_{∞} . Also, for $i, j \in \mathbb{Z}$ we put $[i, j] = \{k \in \mathbb{Z} : i \leq k \leq j\}$ and $[i, \infty) = \{k \in \mathbb{Z} : k \geq i\}$. Thus $\Pi[1, \infty)$ is the set of pole numbers of Q_{∞} , $\Pi(r)$ is the r th pole number, and $\Pi^{-1}[R_1, R_2] = \{r : R_1 \leq \Pi(r) \leq R_2\}$. We note that $\Pi^{-1}[0, R] = \{r : \Pi(r) \leq R\}$ and $\Pi^{-1}[R_1, R_2] = \Pi^{-1}[0, R_2] \setminus \Pi^{-1}[0, R_1 - 1]$. From [14, Proposition VI.4.1] we deduce that

$$(2) \quad \Pi[1, \infty) = \{iq + j : 0 \leq i \leq q - 2, 0 \leq j \leq i\} \cup [2g, \infty).$$

We note that, for $m < n$, $\dim(mQ_{\infty} - D) = 0$ and $k = \dim(mQ_{\infty}) = |\Pi^{-1}[0, m]|$.

State complexity of Hermitian codes. For $0 \leq i \leq n$ we put $D_{i,-} = \sum_{j=1}^i Q_{l_j}$ and $D_{i,+} = \sum_{l=i+1}^n Q_{l_j}$ (where (l_1, \dots, l_n) is a fixed but arbitrary permutation of $(1, \dots, n)$). We deduce that $s_i(C_{\mathcal{L}}(D, mQ_{\infty})) = k - \dim(mQ_{\infty} - D_{i,-}) - \dim(mQ_{\infty} - D_{i,+}) + 2\dim(mQ_{\infty} - D)$. In particular, for $m < n$,

$$(3) \quad s_i(C_{\mathcal{L}}(D, mQ_{\infty})) = k - \dim(mQ_{\infty} - D_{i,-}) - \dim(mQ_{\infty} - D_{i,+}).$$

These identities yield $s(C_{\mathcal{L}}(D, mQ_{\infty})) = W(C_{\mathcal{L}}(D, mQ_{\infty}))$ for $m \in [0, \frac{n-2}{2}] \cup [\frac{n-2}{2} + 2g, n + 2g - 2]$. Thus we will almost exclusively be interested in $m \in I(n, g) = [\frac{n-1}{2}, \frac{n-3}{2} + 2g]$. In fact, since $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ if and only if $m^{\perp} \in [\frac{n-1}{2}, \frac{n-3}{2} + g]$, we will often restrict our attention to $m \in [\frac{n-1}{2}, \frac{n-3}{2} + g]$, deducing results for $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ from $s(C^{\perp}) = s(C)$ and $\nabla(C^{\perp}) = \nabla(C)$.

It is convenient to put $J(n, g) = [\frac{n-1}{2}, \frac{n-2}{2} + g]$. Using results of [11, 16], [5, Proposition 5.1] shows that for $m \in I(n, g)$,

$$(4) \quad \nabla_i(C_{\mathcal{L}}(D, mQ_{\infty})) = k - |\Pi^{-1}[0, m - i]| - |\Pi^{-1}[0, m + i - n]|,$$

which is used to prove the following theorem.

THEOREM 2.1 (see [5, Theorem 5.5]). *For $m \in J(n, g)$, write $n - 2m + 4g + q - 2 = uq + v$, where $0 \leq v \leq q - 1$. Then $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$ is attained at $m - 2g + 1 + \lfloor \frac{u}{2} \rfloor q$ and equals*

$$k - \binom{q - \lfloor \frac{u}{2} \rfloor}{2} - \binom{q - \lceil \frac{u}{2} \rceil}{2} - \min \left\{ q - \lceil \frac{u}{2} \rceil, q - v \right\}.$$

If $C_{\mathcal{L}}(D, mQ_{\infty})$ is DLP-tight, then we just say m is *DLP-tight*.

3. When the DLP bound is not tight. Let $m \in [0, \frac{n-2}{2}] \cup [\frac{n-2}{2} + 2g, n + 2g - 2]$. Then by [5, Proposition 4.3, Example 4.9], we have $\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = W(C_{\mathcal{L}}(D, mQ_{\infty}))$ and so

$$\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = s[C_{\mathcal{L}}(D, mQ_{\infty})] = s(C_{\mathcal{L}}(D, mQ_{\infty})),$$

where $C_{\mathcal{L}}(D, mQ_{\infty})$ can have any coordinate order. Such m are therefore DLP-tight, and we are reduced to determining which $m \in I(n, g)$ are DLP-tight. We note that $\frac{n-3}{3} + 2g < n$, so that the codes that we are interested in are nonabundant.

TABLE 1
Table of new notation.

m	integer
q	fixed prime power
m^\perp	$n + 2g - 2 - m$
q_2	$q \bmod 2$
$I(n, g)$	$[\frac{n-1}{2}, \frac{n-3}{2} + 2g]$
$J(n, g)$	$[\frac{n-1}{2}, \frac{n-2}{2} + g]$
M	$m - \frac{q^2 - q_2}{2} q$ if $m \in J(n, g)$
M^\bullet, M°	$M = M^\bullet(q + 1) + M^\circ$ where $0 \leq M^\circ \leq q$
$\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$	Improved DLP bound for $m \in I(n, g)$ Definition 3.11
$\Delta(m)$	$\nabla^i(C_{\mathcal{L}}(D, mQ_\infty)) - \nabla(C_{\mathcal{L}}(D, mQ_\infty))$ (Theorem 3.9 and Corollary 3.10)
\mathbb{F}_H^1	Finite places of degree one in \mathbb{F}_H
α_{ab}	Elements of \mathbb{F}_{q^2} such that $\alpha_{ab}^{q+1} = a \in \mathbb{F}_q$
β_{ac}	Elements of \mathbb{F}_{q^2} such that $\beta_{ac}^q + \beta_{ac} = a \in \mathbb{F}_q$
$Q_{a,b,c} = Q_{\alpha_{ab}, \beta_{ac}}$	Element of \mathbb{F}_H^1 such that $x(Q_{a,b,c}) = \alpha_{ab}$ and $y(Q_{a,b,c}) = \beta_{ac}$
C_m	Element of $[C_{\mathcal{L}}(D, mQ_\infty)]$ with coordinate order given in section 4
$P_{\text{gain}}(m), P_{\text{fall}}(m)$	Sets of points of gain and fall of C_m
$P_{\text{gain}}^{i,-}(m), P_{\text{fall}}^{i,-}(m)$	$ P_{\text{gain}}(m) \cap [1, i] $ and $ P_{\text{fall}}(m) \cap [1, i] $
Λ	$\Lambda : [0, \infty) \times [0, q - 1] \rightarrow [0, \infty)$ given by $\Lambda(j, l) = jq + l(q + 1)$
ζ_{gain}	$0, \frac{q - q_2}{2}, q$ depending on M° (defined before Proposition 4.8)
ζ_{fall}	$0, \frac{q + q_2}{2}, q$ depending on M° (defined before Proposition 4.8)
$\theta_{\text{gain}}, \theta_{\text{fall}}$	$M^\bullet + M^\circ - \zeta_{\text{gain}}$ and $M^\bullet + M^\circ - \zeta_{\text{fall}}$
ζ_{norm}	$(\zeta_{\text{gain}} + \zeta_{\text{fall}})/2$
η	$2q - 2M^\bullet + q_2 - \zeta_{\text{norm}} - 3$

In this section we determine the $m \in I(n, g)$ which are not DLP-tight, i.e. with $s[C_{\mathcal{L}}(D, mQ_\infty)] > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$. The coordinate order of $C_{\mathcal{L}}(D, mQ_\infty)$ is arbitrary, so it suffices to show that $s(C_{\mathcal{L}}(D, mQ_\infty)) > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$.

Our approach has three steps.

(i) We prove the key lemma, Lemma 3.2, and indicate how this can be used to show that m is not DLP-tight (Example 3.3).

(ii) We prove a generalization of the key lemma (Lemma 3.4) and an application of Proposition 3.5. We indicate how this can be used to improve on the DLP bound by more than one (Example 3.6).

(iii) We prove an application of Proposition 3.5 to improve the DLP bound for $m \in I(n, g)$, Theorem 3.9, and Corollary 3.10.

We conclude section 3 with a table of the improved DLP bound for small values of q (2) and an analysis of the proportion of those $m \in I(n, g)$ for which our bound is strictly better than the DLP bound (Proposition 3.12).

The key lemma. We begin with a clarification of (3) and (4).

LEMMA 3.1. For $0 \leq i \leq n$ and $m \in I(n, g)$,

(5)

$$\dim(mQ_\infty - D_{i,-}) \leq |\Pi^{-1}[0, m - i]| \text{ and } \dim(mQ_\infty - D_{i,+}) \leq |\Pi^{-1}[0, m + i - n]|$$

and $s_i(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla_i(C_{\mathcal{L}}(D, mQ_\infty))$ if and only if there is equality in both.

Proof. The first part follows from [5, Lemma 4.1] and the fact that the gonality sequence of H/\mathbb{F}_{q^2} equals the pole number sequence of Q_∞ by [12, Corollary 2.4]. The second part then follows from (3) and (4). \square

We note that Lemma 3.1 implies that a coordinate order is inefficient, in the sense of [9], if and only if there exists an i , $0 \leq i \leq n$, such that $|\Pi^{-1}[0, m - i]| >$

$\dim(mQ_\infty - D_{i,-})$ or $|\Pi^{-1}[0, m+i-n]| > \dim(mQ_\infty - D_{i,+})$. To show the stronger result that $s(C_{\mathcal{L}}(D, mQ_\infty)) > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$, we require a stronger condition on i , namely, that it satisfies

$$\begin{aligned} |\Pi^{-1}[0, m-i]| - \dim(mQ_\infty - D_{i,-}) + |\Pi^{-1}[0, m+i-n]| - \dim(mQ_\infty - D_{i,+}) \\ > \nabla(C_{\mathcal{L}}(D, mQ_\infty)) - \nabla_i(C_{\mathcal{L}}(D, mQ_\infty)), \end{aligned}$$

so that $s_i(C_{\mathcal{L}}(D, mQ_\infty)) > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$.

This stronger condition is clearly more likely to hold if $\nabla_i(C_{\mathcal{L}}(D, mQ_\infty))$ attains or is close to attaining $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$. For now, we concentrate on determining when the equalities in (5) cannot hold. For these equalities to hold, $\dim(mQ_\infty - D_{i,-})$ and $\dim(mQ_\infty - D_{i,+})$ must change with $|\Pi^{-1}[0, m-i]|$ and $|\Pi^{-1}[0, m+i-n]|$, respectively. We shall see that it is possible that both $|\Pi^{-1}[0, m-i]| = |\Pi^{-1}[0, m-(i-1)]| - 1$ and $|\Pi^{-1}[0, m+i-n]| = |\Pi^{-1}[0, m+(i-1)-n]| + 1$ (i.e., it is possible that both $m-i+1$ and $m+i-n$ are pole numbers of Q_∞).

LEMMA 3.2. *For $m \leq \frac{n-2}{2} + g$, it is not possible that $\dim(mQ_\infty - D_{i,-}) = \dim(mQ_\infty - D_{i-1,-}) - 1$ and $\dim(mQ_\infty - D_{i,+}) = \dim(mQ_\infty - D_{i-1,+}) + 1$.*

Proof. We assume that $\dim(mQ_\infty - D_{i,-}) = \dim(mQ_\infty - D_{i-1,-}) - 1$ and $\dim(mQ_\infty - D_{i,+}) = \dim(mQ_\infty - D_{i-1,+}) + 1$ and derive a contradiction. Suppose we have $z_1, z_2 \in H/\mathbb{F}_{q^2}$ such that (i) $(z_1) \geq -mQ_\infty + D_{i-1,-}$, $v_{Q_{l_i}}(z_1) = 0$ and (ii) $(z_2) \geq -mQ_\infty + D_{i,+}$, $v_{Q_{l_i}}(z_2) = 0$. Thus $(z_1 z_2) \geq -2mQ_\infty + D - Q_{l_i}$ and $v_{Q_{l_i}}(z_1 z_2) = 0$. Now $nQ_\infty - D$ is a principal divisor of H/\mathbb{F}_{q^2} (e.g., as in the proof of [14, Proposition VII.4.2]), say $nQ_\infty - D = (z_3)$. Thus $(z_1 z_2 z_3) \geq (n-2m)Q_\infty - Q_{l_i}$ and $v_{Q_{l_i}}(z_1 z_2 z_3) = -1$. Hence $z_1 z_2 z_3 \in \mathcal{L}((2m-n)Q_\infty + Q_{l_i}) \setminus \mathcal{L}((2m-n)Q_\infty)$ so that by [14, Lemma I.4.8]

$$(6) \quad \dim((2m-n)Q_\infty + Q_{l_i}) = \dim((2m-n)Q_\infty) + 1.$$

Now $(2g-2)Q_\infty$ is a canonical divisor of H/\mathbb{F}_{q^2} (e.g., by [14, Lemma VI.4.4] or because $2g-2$ is the g th pole number of Q_∞ and [14, Proposition I.6.2]). Thus $\dim((2m-n)Q_\infty + Q_{l_i}) = 2m-n+2-g + \dim((2g-2-2m+n)Q_\infty - Q_{l_i})$ by the Riemann–Roch theorem, so from (6), $\dim((2g-2-2m+n)Q_\infty - Q_{l_i}) = \dim((2m-n)Q_\infty) - 2m+n+g-1$. Again, by the Riemann–Roch theorem, $\dim((2g-2-2m+n)Q_\infty) = g-1-2m+n + \dim((2m-n)Q_\infty)$, so that

$$\dim((2g-2-2m+n)Q_\infty - Q_{l_i}) = \dim((2g-2-2m+n)Q_\infty),$$

and hence $\mathcal{L}((2g-2-2m+n)Q_\infty - Q_{l_i}) = \mathcal{L}((2g-2-2m+n)Q_\infty)$. However, for $2g-2-2m+n \geq 0$, i.e., for $m \leq \frac{n-2}{2} + g$, $\mathbb{F}_{q^2} \subseteq \mathcal{L}((2g-2-2m+n)Q_\infty) \setminus \mathcal{L}((2g-2-2m+n)Q_\infty - Q_{l_i})$, giving the required contradiction. \square

EXAMPLE 3.3. *Let $q = 3$. We show $m = 13$ is not DLP-tight. From (2), we have $\Pi[1, 11] = \{0, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13\}$, so that $k = 11$. From Theorem 2.1, $\nabla(C_{\mathcal{L}}(D, mQ_\infty)) = 10$. Now, from (4),*

$$\nabla_{13}(C_{\mathcal{L}}(D, mQ_\infty)) = k - |\Pi^{-1}[0, 0]| - |\Pi^{-1}[0, -1]| = 10 = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$$

and, similarly, $\nabla_{14}(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$. Thus, $s(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$ implies that $s_i(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla_i(C_{\mathcal{L}}(D, mQ_\infty))$ for $i = 13, 14$. Lemma 3.1 then implies that $\dim(mQ_\infty - D_{13,-}) = |\Pi^{-1}[0, 0]| = 1$, $\dim(mQ_\infty - D_{13,+}) = 0$, $\dim(mQ_\infty - D_{14,-}) = 0$, and $\dim(mQ_\infty - D_{14,+}) = 1$, which contradicts Lemma 3.2. Therefore $s(C_{\mathcal{L}}(D, mQ_\infty)) > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$ and since the coordinate

order of $C_{\mathcal{L}}(D, mQ_{\infty})$ is arbitrary, m is not DLP-tight. We will see in section 5 that 14 and 15 are DLP-tight.

Generalization of the key lemma. Since $\dim(mQ_{\infty} - D_{i-1,-}) \leq \dim(mQ_{\infty} - D_{i,-}) + 1$ and $\dim(mQ_{\infty} - D_{i,+}) \leq \dim(mQ_{\infty} - D_{i-1,+}) + 1$ by [14, Lemma I.4.8], Lemma 3.2 can be restated as follows: for $m \leq \frac{n-2}{2} + g$, either $\dim(mQ_{\infty} - D_{i-1,-}) \leq \dim(mQ_{\infty} - D_{i,-})$ or $\dim(mQ_{\infty} - D_{i,+}) \leq \dim(mQ_{\infty} - D_{i-1,+})$. This generalizes as the following lemma.

LEMMA 3.4. For $m \leq \frac{n-2}{2} + g$ and $0 < t \leq i \leq n$, (i) $\dim(mQ_{\infty} - D_{i-t,-}) \leq \dim(mQ_{\infty} - D_{i,-}) + \lfloor \frac{t}{2} \rfloor$ or (ii) $\dim(mQ_{\infty} - D_{i,+}) \leq \dim(mQ_{\infty} - D_{i-t,+}) + \lfloor \frac{t}{2} \rfloor$.

Proof. Suppose that $\dim(mQ_{\infty} - D_{i,-}) < \dim(mQ_{\infty} - D_{i-t,-}) - \lfloor \frac{t}{2} \rfloor$ and $\dim(mQ_{\infty} - D_{i,+}) > \dim(mQ_{\infty} - D_{i-t,+}) + \lfloor \frac{t}{2} \rfloor$. Therefore there are $r, s > \lfloor \frac{t}{2} \rfloor$ and $\{i_1, \dots, i_r\}, \{j_1, \dots, j_s\} \subseteq \{i-t+1, \dots, i\}$ such that $\dim(mQ_{\infty} - D_{i_k,-}) = \dim(mQ_{\infty} - D_{i_{k-1,-})} - 1$ for $1 \leq k \leq r$ and $\dim(mQ_{\infty} - D_{j_k,+}) = \dim(mQ_{\infty} - D_{j_{k-1,+})} + 1$ for $1 \leq k \leq s$. However, $r + s > t$ so that, since $|\{i-t+1, \dots, i\}| = t$, $\{i_1, \dots, i_r\} \cap \{j_1, \dots, j_s\} \neq \emptyset$, contradicting Lemma 3.2. \square

The following application of Lemmas 3.1 and 3.4 is a straightforward consequence of (3), (4).

PROPOSITION 3.5. For $m \in J(n, g)$ and $0 < t \leq i \leq n$,

$$s[C_{\mathcal{L}}(D, mQ_{\infty})] \geq \nabla_i(C_{\mathcal{L}}(D, mQ_{\infty})) + |\Pi^{-1}[m+i-n-t+1, m+i-n]| - \left\lfloor \frac{t}{2} \right\rfloor.$$

EXAMPLE 3.6. Let $q = 7$ and $m = 186$. Then $s[C_{\mathcal{L}}(D, mQ_{\infty})] \geq \nabla(C_{\mathcal{L}}(D, mQ_{\infty})) + 2 = 159$. We have $k = 166$ (e.g., by the Riemann–Roch theorem). From (2), the first few pole numbers of Q_{∞} are $\Pi[1, 6] = \{0, 7, 8, 14, 15, 16\}$. From Theorem 2.1, we have $\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = 157$. For $i = 173$, $\Pi^{-1}[0, m-i] = \{0, 7, 8\}$ and $\Pi^{-1}[0, m+i-n] = \{0, 7, 8, 14, 15, 16\}$, so that, from (4), $\nabla_i(C_{\mathcal{L}}(D, mQ_{\infty})) = 157 = \nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$. Also, with $t = 3$, we have $\Pi^{-1}(m+i-n-t) = \{0, 7, 8\}$. Thus Proposition 3.5 gives

$$s[C_{\mathcal{L}}(D, mQ_{\infty})] \geq \nabla_i(C_{\mathcal{L}}(D, mQ_{\infty})) + 2 = \nabla(C_{\mathcal{L}}(D, mQ_{\infty})) + 2 = 159.$$

We shall see in section 5 that $s[C_{\mathcal{L}}(D, mQ_{\infty})] = 159$.

Improvement on the DLP bound. We show how Proposition 3.5 can be used to improve on the DLP bound generally. First, we introduce some useful notation: $q_2 = 0$ if q is even and $q_2 = 1$ if q is odd. For a fixed $m \in J(n, g)$, we put $M = m - \frac{q^2 - q_2}{2}q$ and write $M = M^{\bullet}(q+1) + M^{\circ}$, where $0 \leq M^{\circ} \leq q$. We easily deduce the following lemma.

LEMMA 3.7. (i) For q odd, $0 \leq M^{\bullet} \leq \frac{q-3}{2}$ and if $M^{\bullet} = 0$, then $M^{\circ} \geq \frac{q-1}{2}$; (ii) for q even, $0 \leq M^{\bullet} \leq \frac{q-2}{2}$ and if $M^{\bullet} = \frac{q-2}{2}$, then $M^{\circ} = 0$.

We begin by reinterpreting Theorem 2.1 in terms of M^{\bullet} and M° .

LEMMA 3.8. For $m \in J(n, g)$, the DLP bound is attained at

$$\begin{array}{ll} m+1 - M^{\bullet}q & \text{if } 0 \leq M^{\circ} \leq \frac{q-2}{2} - M^{\bullet}, \\ m+1 - (M^{\bullet} + 1 - q_2)q & \text{if } \frac{q-1}{2} - M^{\bullet} \leq M^{\circ} \leq q - M^{\bullet} - 1, \\ m+1 - (M^{\bullet} + 1)q & \text{if } q - M^{\bullet} \leq M^{\circ} \leq q. \end{array}$$

Proof. If u, v are defined as in Theorem 2.1, then

$$(u, v) = \begin{cases} (2q-2-2M^{\bullet}+q_2, q-2M^{\bullet}-2M^{\circ}-2) & \text{if } 0 \leq M^{\circ} \leq \frac{q-2}{2} - M^{\bullet}, \\ (2q-3-2M^{\bullet}+q_2, 2q-2M^{\bullet}-2M^{\circ}-2) & \text{if } \frac{q-1}{2} - M^{\bullet} \leq M^{\circ} \leq q - M^{\bullet} - 1, \\ (2q-4-2M^{\bullet}+q_2, 3q-2M^{\bullet}-2M^{\circ}-2) & \text{if } q - M^{\bullet} \leq M^{\circ} \leq q. \end{cases}$$

The result now follows from the fact that the DLP bound is attained at $m - 2g + 1 + \lfloor \frac{u}{2} \rfloor q$. \square

Next we give our improvement on the DLP bounds for $m \in J(n, g)$. The size of the improvement is given by

$$\Delta(m) = \begin{cases} 1 + M^\bullet + M^\circ - \frac{q-q_2}{2} & \text{if } \frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-M^\bullet-1}{2}, \\ \frac{q+q_2}{2} - M^\circ & \text{if } \frac{q-M^\bullet}{2} \leq M^\circ \leq \frac{q-2+q_2}{2}, \\ 1 + M^\bullet + M^\circ - q & \text{if } q - M^\bullet \leq M^\circ \leq q - \frac{M^\bullet+1}{2}, \\ 1 + q - q_2 - M^\circ & \text{if } q - \frac{M^\bullet}{2} \leq M^\circ \leq q - q_2, \\ 0 & \text{otherwise.} \end{cases}$$

We note that $\Delta(m) > 0$ if and only if $\frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-2+q_2}{2}$ or $q - M^\bullet \leq M^\circ \leq q - q_2$.

THEOREM 3.9. *For $m \in J(n, g)$, $s[C_{\mathcal{L}}(D, mQ_\infty)] \geq \nabla(C_{\mathcal{L}}(D, mQ_\infty)) + \Delta(m)$.*

Proof. First assume that $\frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-2+q_2}{2}$. From Lemma 3.8, $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$ is attained at $i = m + 1 - (M^\bullet + 1 - q_2)q$. We take $i = m + 1 - (M^\bullet + 1 - q_2)q$ and $t = 2M^\bullet + 2M^\circ + 1 - q + q_2$ in Proposition 3.5. Now $m + i - t - n = M^\bullet q - q_2$. We have two subcases.

(a) For $\frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-M^\bullet-1}{2}$ we have $0 < t \leq M^\bullet + q_2$. Now, from (2), $M^\bullet q, \dots, M^\bullet q + M^\bullet \in \Pi[1, \infty)$, so that $|\Pi^{-1}[m + i - n - t + 1, m + i - n]| = t$, and Proposition 3.5 gives

$$s[C_{\mathcal{L}}(D, mQ_\infty)] - \nabla(C_{\mathcal{L}}(D, mQ_\infty)) \geq \left\lceil \frac{t}{2} \right\rceil = 1 + M^\bullet + M^\circ - \frac{q - q_2}{2}.$$

(b) For $\frac{q-M^\bullet}{2} \leq M^\circ \leq \frac{q-2+q_2}{2}$ we have $M^\bullet + q_2 + 1 \leq t \leq 2M^\bullet + 2q_2 - 1 \leq q - 1$. From (2), $M^\bullet q + M^\bullet + 1, \dots, M^\bullet q + q - 1 \notin \Pi(\mathbb{N})$ since $M^\bullet \leq q - 2$, so that $|\Pi^{-1}[m + i - n - t + 1, m + i - n]| = M^\bullet + q_2$, and Proposition 3.5 gives

$$s[C_{\mathcal{L}}(D, mQ_\infty)] - \nabla(C_{\mathcal{L}}(D, mQ_\infty)) \geq M^\bullet + q_2 - \left(M^\bullet + M^\circ - \frac{q - q_2}{2} \right) = \frac{q + q_2}{2} - M^\circ.$$

Now suppose that $q - M^\bullet \leq M^\circ \leq q - q_2$. From Lemma 3.8, $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$ is attained at $m + 1 - (M^\bullet + 1)q$. We take $i = m + 1 - (M^\bullet + 1)q$ and $t = 2M^\bullet + 2M^\circ - 2q + 2 - q_2$ in Proposition 3.5. Now $m + i - t - n = (M^\bullet + 1 - q_2)q - (1 - q_2)$ and again we have two subcases.

(a) For $q - M^\bullet \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$ we have $0 < t \leq M^\bullet + 1 - q_2$. From (2), $(M^\bullet + 1 - q_2)q, \dots, (M^\bullet + 1 - q_2)q + (M^\bullet + 1 - q_2) \in \Pi[1, \infty)$, so that $|\Pi^{-1}[m + i - n - t + 1, m + i - n]| = t$, and Proposition 3.5 gives $s[C_{\mathcal{L}}(D, mQ_\infty)] - \nabla(C_{\mathcal{L}}(D, mQ_\infty)) \geq \left\lceil \frac{t}{2} \right\rceil = 1 + M^\bullet + M^\circ - q$.

(b) For $q - \frac{M^\bullet}{2} \leq M^\circ \leq q - q_2$ we have $M^\bullet + 2 - q_2 \leq t \leq 2M^\bullet + 2 - 3q_2 \leq q - q_2$. From (2), $(M^\bullet + 1 - q_2)q + (M^\bullet + 2 - q_2), \dots, (M^\bullet + 1 - q_2)q + (q - 1) \notin \Pi[1, \infty)$, since $M^\bullet + 1 - q_2 \leq q - 2$, so that $|\Pi^{-1}[m + i - n - t + 1, m + i - n]| = M^\bullet + 2 - 2q_2$, so that from Proposition 3.5,

$$\begin{aligned} s[C_{\mathcal{L}}(D, mQ_\infty)] - \nabla(C_{\mathcal{L}}(D, mQ_\infty)) &\geq M^\bullet + 2 - 2q_2 - (M^\bullet + M^\circ - q + 1 - q_2) \\ &= 1 + q - q_2 - M^\circ. \quad \square \end{aligned}$$

For $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ we put $\Delta(m) = \Delta(m^\perp) \geq 0$.

COROLLARY 3.10. *For $m \in I(n, g)$, $s[C_{\mathcal{L}}(D, mQ_\infty)] \geq \nabla(C_{\mathcal{L}}(D, mQ_\infty)) + \Delta(m)$.*

Proof. The proof is an easy consequence of Theorem 3.9, $\nabla(C) = \nabla(C^\perp)$, and the definition of $\Delta(m)$. \square

DEFINITION 3.11. For $m \in I(n, g)$, we put $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla(C_{\mathcal{L}}(D, mQ_\infty)) + \Delta(m)$.

We note that for $m \in I(n, g)$,

$$(7) \quad \nabla^i(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla^i(C_{\mathcal{L}}(D, m^\perp Q_\infty)).$$

In Table 2 we have written $\nabla^i(m)$ for $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$, and the DLP bound is calculated using Theorem 2.1. The bold face entries are those for which $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty)) > \nabla(C_{\mathcal{L}}(D, mQ_\infty))$. (The values of $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ for $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ can of course be deduced from (7).)

TABLE 2
 $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ for $q \in \{2, 3, 4, 5, 7, 8\}$ and $m \in J(n, g)$.

$\frac{q}{2}$	m	4													
	$\nabla^i(m)$	3													
3	m	13	14	15											
	$\nabla^i(m)$	11	11	11											
4	m	32	33	34	35	36	37								
	$\nabla^i(m)$	26	27	27	28	28	28								
5	m	62	63	64	65	66	67	68	69	70	71				
	$\nabla^i(m)$	53	53	54	54	55	56	56	56	56	56				
7	m	171	172	173	174	175	176	177	178	179	180				
	$\nabla^i(m)$	151	151	152	153	153	154	155	156	156	156				
	m	181	182	183	184	185	186	187	188	189	190	191			
	$\nabla^i(m)$	157	157	157	158	159	159	159	159	159	159	159			
8	m	256	257	258	259	260	261	262	263	264	265	266	267	268	269
	$\nabla^i(m)$	228	229	230	231	231	232	233	234	234	234	235	236	236	236
	m	270	271	272	273	274	275	276	277	278	279	280	281	282	283
	$\nabla^i(m)$	237	238	238	238	238	239	239	239	239	240	240	240	240	240

We conclude this section by calculating the proportion of $m \in I(n, g)$ for which $\Delta(m) > 0$.

PROPOSITION 3.12.

$$|\Delta^{-1}(0, \infty)|/|I(n, g)| = \begin{cases} \frac{1}{2} - \frac{1}{2q} & \text{if } q \text{ is odd,} \\ \frac{1}{2} - \frac{3q-5}{2(q^2-q-1)} & \text{if } q \text{ is even.} \end{cases}$$

Proof. We note first that $|I(n, g)| = 2g + q_2 - 1$. Recall from the definition of $\Delta(m)$ that

$$\Delta^{-1}(0, \infty) \cap \left\{ \frac{n-1}{2}, \dots, \frac{n-2}{2} + g \right\} = \left\{ m : \frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-2+q_2}{2} \text{ or } q - M^\bullet \leq M^\circ \leq q - q_2 \right\}.$$

Next we note that $|\Delta^{-1}(0, \infty)| = 2|\Delta^{-1}(0, \infty) \cap J(n, g)|$. This follows from the definition of $\Delta(m)$ for $\frac{n-1}{2} + g \leq m \leq \frac{n-3}{2} + 2g$ when q is odd and from $\frac{n-2}{2} + g \notin \Delta^{-1}(0, \infty)$ when q is even. Now, fixing $0 \leq M^\bullet \leq \frac{q-3}{2}$, we have

$$\left| \left\{ M^\circ : \frac{q-q_2}{2} - M^\bullet \leq M^\circ \leq \frac{q-2+q_2}{2} \text{ or } q - M^\bullet \leq M^\circ \leq q - q_2 \right\} \right| = 2M^\bullet + 1.$$

We note that the restriction $M^\circ \geq \frac{q-1}{2}$ for q odd and $M^\bullet = 0$ from Lemma 3.7 does not affect this. We also note that for q even and $M^\bullet = \frac{q-2}{2}$, the restriction of $M^\circ = 0$ in Lemma 3.7 gives $|\{M^\circ : 1 \leq M^\circ \leq \frac{q-2}{2} \text{ or } q - \frac{q-2}{2} \leq M^\circ \leq q\}| = 0$.

Thus the result follows from

$$|\Delta^{-1}(0, \infty)| = \begin{cases} 2 \sum_{M^\bullet=0}^{\frac{q-3}{2}} (2M^\bullet + 1) = (q-1) + 4\left(\frac{q-1}{2}\right) = \frac{(q-1)^2}{2} & \text{if } q \text{ is odd,} \\ 2 \sum_{M^\bullet=0}^{\frac{q-4}{2}} (2M^\bullet + 1) = (q-2) + 4\left(\frac{q-2}{2}\right) = \frac{(q-2)^2}{2} & \text{if } q \text{ is even.} \end{cases} \quad \square$$

Thus, for large q at least, $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ improves on $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$ for just under half the $m \in I(n, g)$. We shall see in section 5 that m is DLP-tight when $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ fails to improve on $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$.

4. A good coordinate order. We describe a “good” coordinate order for Hermitian codes, denoting the code in $[C_{\mathcal{L}}(D, mQ_\infty)]$ with this coordinate order by C_m . After recalling the notions of points of gain and fall for a linear code, we give the most natural description of the points of gain and fall of C_m in Propositions 4.2 and 4.4. We conclude by characterizing the points of gain and fall of C_m as “runs” in Theorem 4.10 (which we will use in section 5 to derive a formula for $s(C_m)$).

The good coordinate order. As noted at the beginning of section 3, for $m \leq \frac{n-2}{2}$ or $m \geq \frac{n-2}{2} + 2g$, all coordinate orders of $C_{\mathcal{L}}(D, mQ_\infty)$ are equally bad with regard to state complexity. Thus we are interested in $m \in I(n, g)$.

Recall that H/\mathbb{F}_{q^2} has $n+1$ places of degree one, namely Q_∞ , and the finite places of degree one, Q_1, \dots, Q_n . We put $\mathbb{P}_H^1 = \{Q_1, \dots, Q_n\}$. Now

$$C_{\mathcal{L}}(D, mQ_\infty) = \{(z(Q_{l_1}), \dots, z(Q_{l_n})) : z \in \mathcal{L}(mQ_\infty)\}$$

for some fixed but arbitrary ordering $(Q_{l_1}, \dots, Q_{l_n})$ of \mathbb{P}_H^1 . Thus the order of \mathbb{P}_H^1 determines the coordinate order of $C_{\mathcal{L}}(D, mQ_\infty)$. As in [14], for each $(\alpha, \beta) \in \mathbb{F}_{q^2} \times \mathbb{F}_{q^2}$ such that $\beta^q + \beta = \alpha^{q+1}$, there exists a unique $Q_{\alpha\beta} \in \mathbb{P}_H^1$ such that $x(Q_{\alpha\beta}) = \alpha$ and $y(Q_{\alpha\beta}) = \beta$.

We now describe an order of \mathbb{P}_H^1 giving $C_m \in [C_{\mathcal{L}}(D, mQ_\infty)]$. First we relabel the elements of \mathbb{P}_H^1 as $Q_{a,b,c}$ for certain integers a, b, c . We write $\{0, 1, \dots, q-1\}$ for \mathbb{F}_q , where $0 = 0_{\mathbb{F}_q}$. Now for each $a \in \mathbb{F}_q \setminus \{0\}$ there exist $\beta_{a0}, \dots, \beta_{a,q-1} \in \mathbb{F}_{q^2}$ and $\alpha_{a0}, \dots, \alpha_{aq} \in \mathbb{F}_{q^2}$ such that $\beta_{ac}^q + \beta_{ac} = \alpha_{ab}^{q+1} = a$ for $0 \leq c \leq q-1$ and $0 \leq b \leq q$. Thus for each $a \in \mathbb{F}_q \setminus \{0\}$, $0 \leq c \leq q-1$ and $0 \leq b \leq q$, there exists $Q_{a,b,c} \in \mathbb{P}_H^1$ such that $x(Q_{a,b,c}) = \alpha_{a,b}$ and $y(Q_{a,b,c}) = \beta_{a,c}$, giving $q^3 - q$ elements of \mathbb{P}_H^1 .

For $a = 0$ there exist $\beta_{00}, \dots, \beta_{0q}$ and $\alpha_{00} = 0$ such that $\beta_{0c}^q + \beta_{0c} = \alpha_{00}^{q+1} = 0$ for $0 \leq c \leq q-1$. Thus the remaining q elements of \mathbb{P}_H^1 , which we write as $Q_{0,0,c}$ for $0 \leq c \leq q-1$, are such that $x(Q_{0,0,c}) = 0$ and $y(Q_{0,0,c}) = \beta_{0,c}$. We note that $Q_{a,b,c} = Q_{\alpha_{a,b}, \beta_{a,c}}$.

When a, b , or c takes any of its possible values we write $Q_{*,b,c}$, $Q_{a,*,c}$, or $Q_{a,b,*}$. Note that for $a = 0$ we have $b = 0$ and for $1 \leq a \leq q-1$ we have $0 \leq b \leq q$. Thus there are q places of the form $Q_{0,*,*}$ and for $1 \leq a \leq q-1$ there are $q^2 - 1$ places of the form $Q_{a,*,*}$.

We first describe the ordering of \mathbb{P}_H^1 giving $C_m \in [C_{\mathcal{L}}(D, mQ_\infty)]$ for $m \in J(n, g)$. This uses lexicographic order of t -tuples of integers: $(i_1, \dots, i_t) < (j_1, \dots, j_t)$ if and

only if there exists u such that $i_1 = j_1, \dots, i_{u-1} = j_{u-1}$ and $i_u < j_u$. For $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$ or $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$, C_m is defined by simply using the order

$$\text{O1: } Q_{a,b,c} < Q_{a',b',c'} \text{ if } (a,b,c) < (a',b',c')$$

of \mathbb{P}_H^1 . For $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$, C_m is defined by the ‘‘Order O2’’ of \mathbb{P}_H^1 : partition \mathbb{P}_H^1 into the following three sets:

$$(8) \quad \begin{aligned} \mathbb{P}_1^1 &= \{Q_{1,*,c} : 0 \leq c \leq \frac{q-q_2}{2} - 1\}, \\ \mathbb{P}_2^1 &= \{Q_{a,*,*} : a \neq 1\}, \\ \mathbb{P}_3^1 &= \{Q_{1,*,c} : \frac{q-q_2}{2} \leq c \leq q - 1\}. \end{aligned}$$

Then Order O2 of \mathbb{P}_H^1 is given by putting $P_1^1 < P_2^1 < P_3^1$, ordering \mathbb{P}_1^1 and \mathbb{P}_3^1 by $Q_{1,b,c} < Q_{1,b',c'}$ if $(c,b) < (c',b')$, and ordering \mathbb{P}_2^1 by $Q_{a,b,c} < Q_{a',b',c'}$ if $(a,b,c) < (a',b',c')$.

For $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$, the coordinate order of C_m is defined to be that of C_{m^\perp} .

From now on, Q_i denotes the i th element of \mathbb{P}_H^1 ordered as above. Thus

$$C_m = \{(z(Q_1), \dots, z(Q_n)) : z \in \mathcal{L}(mQ_\infty)\}.$$

The points of gain and fall of C_m . Points of gain and fall were introduced in [3, 6]. For this paragraph, C is a length n linear code with dimension k . We note that $\dim(C_{i,-})$ (as defined in section 2) increases in unit steps from 0 to k and $\dim(C_{i,+})$ decreases in unit steps from k to 0 as i increases from 0 to n . If $0 \leq i \leq n$, then

- i is a point of gain of C if $\dim(C_{i,+}) = \dim(C_{i-1,+}) - 1$ and
- i is a point of fall of C if $\dim(C_{i,-}) = \dim(C_{i-1,-}) + 1$.

These definitions are motivated by (1). We note that there are k points of gain and k points of fall. Points of gain and fall describe the local behavior of a minimal trellis [6], and being able to give a succinct characterization of them for particular families of codes has been useful in calculating formulae for their state complexity; see, e.g., [3, 6]. The same proves to be the case here. We note that, as in [6], i is a point of gain of C_m if and only if i is the ‘‘initial point’’ of a codeword of C_m , i.e., if and only if there exists $z \in \mathcal{L}(mQ_\infty)$ such that

$$z(Q_1) = \dots = z(Q_{i-1}) = 0 \text{ and } z(Q_i) \neq 0.$$

Similarly, i is a point of fall of C_m if and only if i is the ‘‘final point’’ of a codeword of C_m , i.e., if and only if there exists $z \in \mathcal{L}(mQ_\infty)$ such that

$$z(Q_i) \neq 0 \text{ and } z(Q_{i+1}) = \dots = z(Q_n) = 0.$$

We write $P_{\text{gain}}(C)$ and $P_{\text{fall}}(C)$ for the sets of points of gain and fall of C . With $P_{\text{gain}}^{i,-}(C) = |P_{\text{gain}}(C) \cap [1, i]|$ and $P_{\text{fall}}^{i,-}(C) = |P_{\text{fall}}(C) \cap [1, i]|$ we have

$$(9) \quad s_i(C) = P_{\text{gain}}^{i,-}(C) - P_{\text{fall}}^{i,-}(C).$$

We also write $P_{\text{gain}}(m) := P_{\text{gain}}(C_m)$ and $P_{\text{fall}}(m) := P_{\text{fall}}(C_m)$. We will need a function Λ closely related to Π . Define $\Lambda : [0, \infty) \times [0, q-1] \rightarrow [0, \infty)$ by

$$\Lambda(j, l) = jq + l(q+1).$$

We have $\Pi[1, \infty) = \text{Im}(\Lambda)$ from [14]. We note that

$$\Lambda^{-1}[0, m] = \{(j, l) \in \mathbb{Z} \times \mathbb{Z} : j \geq 0, 0 \leq l \leq q-1, jq + l(q+1) \leq m\}$$

and for $m < n$, $k = \dim(C_m) = |\Lambda^{-1}[0, m]|$ [14, Proposition VII.4.3]. For $0 \leq a \leq q-1$, we put

$$A(a) = \begin{cases} \{\alpha_{00}\} & \text{for } a = 0, \\ \{\alpha_{ab} : 0 \leq b \leq q\} & \text{for } 1 \leq a \leq q-1 \end{cases}$$

and $B(a) = \{\beta_{ac} : 0 \leq c \leq q-1\}$. Thus $\mathbb{P}_H^1 = \{Q_{\alpha\beta} : 0 \leq a \leq q-1, \alpha \in A(a), \beta \in B(a)\}$. We also put $A = \bigcup_{a=0}^{q-1} A(a)$ and $B = \bigcup_{a=0}^{q-1} B(a)$. We will determine the initial and final points of certain $z \in H/\mathbb{F}_{q^2}$ of the form

$$z = (x - \alpha_0) \cdots (x - \alpha_{l-1})(y - \beta_0) \cdots (y - \beta_{j-1}),$$

where $\alpha_0, \dots, \alpha_{l-1} \in A$ and $\beta_0, \dots, \beta_{j-1} \in B$. Note that $(x - \alpha_{ab})(Q_{a',b',*}) = 0$ if and only if $a = a'$, $b = b'$ and $(y - \beta_{ac})(Q_{a',*,c'}) = 0$ if and only if $a = a'$, $c = c'$. Of course, we are interested in when $(z(Q_1), \dots, z(Q_n)) \in C_m$, i.e., when $z \in \mathcal{L}(mQ_\infty)$.

LEMMA 4.1. *If $(j, l) \in \Lambda^{-1}[0, m]$, $\alpha_0, \dots, \alpha_{j-1} \in A$ and $\beta_0, \dots, \beta_{l-1} \in B$, then*

$$(x - \alpha_0) \cdots (x - \alpha_{j-1})(y - \beta_0) \cdots (y - \beta_{l-1}) \in \mathcal{L}(mQ_\infty).$$

Proof. We put $z_{jl} = (x - \alpha_0) \cdots (x - \alpha_{j-1})(y - \beta_0) \cdots (y - \beta_{l-1}) \in \mathcal{L}(mQ_\infty)$. Using the facts that (i) $v_{Q_\infty}(x) = -q$ and $v_{Q_\infty}(y) = -(q+1)$, (ii) for $Q \in \mathbb{P}_H \setminus \{Q_\infty\}$, $v_Q(x) \geq 0$ and $v_Q(y) \geq 0$, and (iii) for $\alpha \in \mathbb{F}_{q^2}$ and $Q \in \mathbb{P}_H$, $v_Q(\alpha) = 0$, we get $v_{Q_\infty}(z_{jl}) = -\Lambda(j, l)$ and $v_Q(z_{jl}) \geq 0$ for all $Q \in \mathbb{P}_H \setminus \{Q_\infty\}$. Hence $(j, l) \in \Lambda^{-1}[0, m]$ implies that $z_{jl} \in \mathcal{L}(mQ_\infty)$. \square

PROPOSITION 4.2 (O1 ordering of \mathbb{P}_H^1). *For $m \in J(n, g)$ with $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$ or $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$,*

1. $P_{\text{gain}}(m) = \{jq + l + 1 : (j, l) \in \Lambda^{-1}[0, m]\}$ and
2. $P_{\text{fall}}(m) = \{n - jq - l : (j, l) \in \Lambda^{-1}[0, m]\} = n - P_{\text{gain}}(m) + 1$.

Proof. We order the set A by $\alpha_{ab} < \alpha_{a'b'}$ if and only if $(a, b) < (a', b')$. Thus $\alpha_{ab} < \alpha_{a'b'}$ if and only if $Q_{a,b,*} < Q_{a',b',*}$. For $0 \leq d \leq q^2 - 1$, we write α_d for the $(d+1)$ st element of A . Thus $\alpha_0 = \alpha_{00}, \alpha_1 = \alpha_{10}, \dots, \alpha_{q+1} = \alpha_{1q}, \dots, \alpha_{q^2-1} = \alpha_{q-1,q}$. For $0 \leq d \leq q^2 - 1$, we define $a(d)$ by $\alpha_{a(d)b} = \alpha_d$ for some b . Thus $a(0) = 0, a(1) = \dots = a(q+1) = 1, \dots, a(q^2 - q - 1) = \dots = a(q^2 - 1) = q - 1$. Then for $1 \leq i \leq q^3$, writing $i - 1 = dq + c$, where $0 \leq d \leq q^2 - 1$ and $0 \leq c \leq q - 1$, we have

$$Q_i = Q_{\alpha_d, \beta_{a(d)c}}.$$

Thus

$$(x - \alpha_d)(Q_i) = 0 \text{ if and only if } dq + 1 \leq i \leq (d+1)q$$

and

$$(y - \beta_{ac})(Q_i) = 0 \text{ if and only if } i = dq + c + 1, \text{ where } a(d) = a.$$

We begin with $P_{\text{gain}}(m)$. For $(j, l) \in \Lambda^{-1}[0, m]$ we put

$$u_j^{\text{gain}} = (x - \alpha_0) \cdots (x - \alpha_{j-1}), v_{jl}^{\text{gain}} = (y - \beta_{a(j),0}) \cdots (y - \beta_{a(j),l-1}), z_{jl}^{\text{gain}} = u_j^{\text{gain}} v_{jl}^{\text{gain}}.$$

We note that $jq \leq \Lambda(j, l) \leq m \leq \frac{n-2}{2} + g = \frac{q^3+q^2-q-2}{2}$, which implies that $j < \frac{q^2+q-1}{2} \leq q^2$, so that $u_j^{\text{gain}}, v_{jl}^{\text{gain}}$, and z_{jl}^{gain} are well-defined for all $(j, l) \in \Lambda^{-1}[0, m]$. Now $u_j^{\text{gain}}(Q_i) = 0$ if and only if $1 \leq i \leq jq$, $v_{jl}^{\text{gain}}(Q_i) = 0$ for $jq+1 \leq i \leq jq+l$, and $v_{jl}^{\text{gain}}(Q_{jq+l+1}) \neq 0$. Hence the initial point of z_{jl}^{gain} is $jq+l+1$ so that $jq+l+1 \in P_{\text{gain}}(m)$. Also, by Lemma 4.1, $z_{jl}^{\text{gain}} \in \mathcal{L}(mQ_\infty)$. Finally, each $(j, l) \in \Lambda^{-1}[0, m]$ gives a different point of gain of C_m and, since $|\Lambda^{-1}[0, m]| = k$, these are all the points of gain, and similarly for points of fall. \square

We use Proposition 4.2 to determine $s(C_m)$ for $q = 2$ and $m \in [\frac{n-1}{2}, \frac{n-3}{2} + g]$. To do this we use (9) and so we put

$$P_{\text{gain}}^{i,-}(m) := P_{\text{gain}}^{i,-}(C_m) \text{ and } P_{\text{fall}}^{i,-}(m) := P_{\text{fall}}^{i,-}(C_m).$$

EXAMPLE 4.3. *If $q = 2$, then $P_{\text{gain}}(4) = [1, 3] \cup \{5\}$ and $P_{\text{fall}}(4) = \{4\} \cup [6, 8]$, giving $s(C_4) = 3$. (Thus C_4 is our first example of a geometric Goppa code with $s(C_4) < W(C_4)$). Also, $s(C_4) = \nabla(C_4)$, where the latter is given by Theorem 2.1.)*

Proof. The coordinate order of C_4 is $Q_{0,0,0} < Q_{0,0,1} < Q_{1,0,0} < Q_{1,0,1} < Q_{1,1,0} < Q_{1,1,1} < Q_{1,2,0} < Q_{1,2,1}$. In the notation of Proposition 4.2, we have $\alpha_0 = \alpha_{0,0}$, $\alpha_1 = \alpha_{1,0}$, $\alpha_2 = \alpha_{1,1}$, $\alpha_3 = \alpha_{1,2}$. Thus $a(0) = 0$ and $a(1) = a(2) = a(3) = 1$. Also $\Lambda^{-1}[0, 4] = \{(0, 0), (1, 0), (0, 1), (2, 0)\}$, and $k = 4$.

Now $P_{\text{gain}}(4)$ is the set of initial points of z_{jl}^{gain} , where $(j, l) \in \Lambda^{-1}[0, 4]$. These are given in the table below. The third column in the table gives the ‘‘initial place,’’ i.e., the $Q_{a,b,c}$, such that $Q_{a,b,c} = Q_i$, where i is the initial point.

(j, l)	z_{jl}^{gain}	Initial place	Initial point
$(0, 0)$	1	$Q_{0,0,0}$	1
$(1, 0)$	$(x - \alpha_0)$	$Q_{1,0,0}$	3
$(0, 1)$	$(y - \beta_{0,0})$	$Q_{0,0,1}$	2
$(2, 0)$	$(x - \alpha_0)(x - \alpha_1)$	$Q_{1,1,0}$	5

Thus $P_{\text{gain}}(4) = [1, 3] \cup \{5\}$. Also $P_{\text{fall}}(4)$ is given by the final points of z_{jl}^{fall} such that $(j, l) \in \Lambda^{-1}[0, 4]$, as follows.

(j, l)	z_{jl}^{fall}	Final place	Final point
$(0, 0)$	1	$Q_{1,2,1}$	8
$(1, 0)$	$(x - \alpha_3)$	$Q_{1,1,1}$	6
$(0, 1)$	$(y - \beta_{1,1})$	$Q_{1,2,0}$	7
$(2, 0)$	$(x - \alpha_2)(x - \alpha_3)$	$Q_{1,0,1}$	4

Thus $P_{\text{fall}}(4) = \{4\} \cup [6, 8]$. Hence, using (9) we have

i	0	1	2	3	4	5	6	7	8
$P_{\text{gain}}^{i,-}(4)$	0	1	2	3	3	4	4	4	4
$P_{\text{fall}}^{i,-}(4)$	0	0	0	0	1	1	2	3	4
$s_i(C_4)$	0	1	2	3	2	3	2	1	0

giving $s(C_4) = 3$. \square

For $m \in J(n, g)$ such that $\frac{q-M^*-1}{2} \leq M^\circ \leq q - \frac{M^*+1}{2}$ we put

$$\zeta_{\text{gain}} = \frac{q - q_2}{2} \text{ and } \zeta_{\text{fall}} = \frac{q + q_2}{2}.$$

PROPOSITION 4.4 (O2 ordering of \mathbb{P}_H^1). For $m \in J(n, g)$ with $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$, $P_{\text{gain}}(m) = P_{\text{gain}}^1(m) \cup P_{\text{gain}}^2(m)$ and $P_{\text{fall}}(m) = P_{\text{fall}}^1(m) \cup P_{\text{fall}}^2(m)$, where

$$\begin{aligned} P_{\text{gain}}^1(m) &= \{l(q+1) + j + 1 : (j, l) \in \Lambda^{-1}[0, m], 0 \leq j \leq q, 0 \leq l \leq \zeta_{\text{gain}} - 1\}, \\ P_{\text{gain}}^2(m) &= \{\zeta_{\text{gain}}(q+1) + jq + l + 1 : (j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q+1)]\}, \\ P_{\text{fall}}^1(m) &= \{n - l(q+1) - j : (j, l) \in \Lambda^{-1}[0, m], 0 \leq j \leq q, 0 \leq l \leq \zeta_{\text{fall}} - 1\}, \\ P_{\text{fall}}^2(m) &= \{n - \zeta_{\text{fall}}(q+1) - jq - l : (j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{fall}}(q+1)]\}. \end{aligned}$$

Proof. We recall that P_1^1 , P_2^1 , and P_3^1 were defined in (8). We note that

- for $1 \leq i \leq \zeta_{\text{gain}}(q+1)$, $Q_i \in \mathbb{P}_1^1$, so that writing $i-1 = c(q+1) + b$, where $0 \leq c \leq \zeta_{\text{gain}} - 1$ and $0 \leq b \leq q$, $Q_i = Q_{1,b,c}$;
- for $\zeta_{\text{gain}}(q+1) + 1 \leq i \leq \zeta_{\text{gain}}(q+1) + q^3 - q^2 - q$, $Q_i \in \mathbb{P}_2^1$; and
- for $\zeta_{\text{gain}}(q+1) + q^3 - q^2 - q \leq i \leq q^3$, $Q_i \in \mathbb{P}_3^1$.

We begin by showing that $P_{\text{gain}}^1(m) \subseteq P_{\text{gain}}(m)$. For $(j, l) \in \Lambda^{-1}[0, m]$ such that $0 \leq j \leq q$ and $0 \leq l \leq \zeta_{\text{gain}} - 1$ we exhibit an element of $\mathcal{L}(mQ_\infty)$ with initial point $l(q+1) + j + 1$. Put

$$u_j^{\text{gain}} = (x - \alpha_{1,0}) \cdots (x - \alpha_{1,j-1}), \quad v_l^{\text{gain}} = (y - \beta_{1,0}) \cdots (y - \beta_{1,l-1}), \quad z_{jl}^{\text{gain}} = u_j^{\text{gain}} v_l^{\text{gain}}.$$

Thus $v_l^{\text{gain}}(Q_{a,*,c}) = 0$ if and only if $a = 1$, and $0 \leq c \leq l-1$ and $u_j^{\text{gain}}(Q_{a,b,*}) = 0$ if and only if $a = 1$, and $0 \leq b \leq j-1$. Therefore $v_l^{\text{gain}}(Q_i) = 0$ if and only if $1 \leq i \leq l(q+1)$, $u_j^{\text{gain}}(Q_i) = 0$ for $l(q+1) + 1 \leq i \leq l(q+1) + j$ (taking $c = l \leq \zeta_{\text{gain}}$), and $u_j^{\text{gain}}(Q_{l(q+1)+j+1}) \neq 0$ (taking $c = l$ and $b = j \leq q$). Hence the initial point of z_{jl}^{gain} is $l(q+1) + j + 1$. Also, from Lemma 4.1, $z_{jl}^{\text{gain}} \in \mathcal{L}(mQ_\infty)$, so that $P_{\text{gain}}^1(m) \subseteq P_{\text{gain}}(m)$.

Next we show that $P_{\text{gain}}^2(m) \subseteq P_{\text{gain}}(m)$. We order $A \setminus A(1)$ by $\alpha_{ab} < \alpha_{a'b'}$ if and only if $(a, b) < (a', b')$ and write α_d for the $(d+1)$ st element of $A \setminus A(1)$, where $0 \leq d \leq q^2 - q - 2$. (This is different from the labelling in the proof of Proposition 4.2 since we do not include $A(1)$ in the relabelling.) We define $a(d)$ by $\alpha_{a(d)b} = \alpha_d$ for some b . Then, for $\zeta_{\text{gain}}(q+1) + 1 \leq i \leq \zeta_{\text{gain}}(q+1) + q^3 - q^2 - q$, writing $i-1 = \zeta_{\text{gain}}(q+1) + dq + c$, where $0 \leq d \leq q^2 - q - 2$ and $0 \leq c \leq q-1$, we have $Q_i = Q_{\alpha_d, \beta_{a(d)c}}$. We put $w^{\text{gain}} = (y - \beta_{1,0}) \cdots (y - \beta_{1, \zeta_{\text{gain}}-1})$. For $(j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q+1)]$, set

$$\begin{aligned} (u^{\text{gain}})'_j &= (x - \alpha_0) \cdots (x - \alpha_{j-1}), \quad (v^{\text{gain}})'_{jl} = (y - \beta_{a(j),0}) \cdots (y - \beta_{a(j),l-1}), \\ z_{j,l+\zeta_{\text{gain}}}^{\text{gain}} &= w^{\text{gain}} (u^{\text{gain}})'_j (v^{\text{gain}})'_{jl}. \end{aligned}$$

We note that $jq \leq \Lambda(j, l) \leq m - \zeta_{\text{gain}}(q+1) \leq \frac{q^3 - q - 1}{2}$, which implies that $j \leq q^2 - q - 2$. Thus $(u^{\text{gain}})'_j$, $(v^{\text{gain}})'_{jl}$, and $z_{j,l+\zeta_{\text{gain}}}^{\text{gain}}$ are well-defined for all $(j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q+1)]$. Now $w^{\text{gain}}(Q_i) = 0$ if and only if $1 \leq i \leq \zeta_{\text{gain}}(q+1)$. Also $(u^{\text{gain}})'_j(Q_{\alpha_d \beta_{a(d)c}}) = 0$ if and only if $0 \leq d \leq j-1$ and $0 \leq c \leq q-1$, and $(v^{\text{gain}})'_{jl}(Q_{\alpha_d \beta_{a(d)c}}) = 0$ if and only if $a(d) = a(j)$ and $0 \leq c \leq l-1$. Thus $(u^{\text{gain}})'_j(Q_i) = 0$ if and only if $\zeta_{\text{gain}}(q+1) + 1 \leq i \leq \zeta_{\text{gain}}(q+1) + jq$, $(v^{\text{gain}})'_{jl}(Q_i) = 0$ for $\zeta_{\text{gain}}(q+1) + jq + 1 \leq i \leq \zeta_{\text{gain}}(q+1) + jq + l$ and $(v^{\text{gain}})'_{jl}(Q_{\zeta_{\text{gain}}(q+1)+jq+l+1}) \neq 0$. Therefore the initial point of $z_{j,l+\zeta_{\text{gain}}}^{\text{gain}}$ is $\zeta_{\text{gain}}(q+1) + jq + l + 1$. Also, by Lemma 4.1, $w^{\text{gain}} \in \mathcal{L}(\zeta_{\text{gain}}(q+1)Q_\infty)$ and $(z_{j,l+\zeta_{\text{gain}}}^{\text{gain}}/w) \in \mathcal{L}((m - \zeta_{\text{gain}}(q+1))Q_\infty)$. Hence $z_{j,l+\zeta_{\text{gain}}}^{\text{gain}} \in \mathcal{L}(mQ_\infty)$, $P_{\text{gain}}^2 \subseteq P_{\text{gain}}(m)$, and $P_{\text{gain}}^1(m) \cup P_{\text{gain}}^2(m) \subseteq P_{\text{gain}}(m)$.

Therefore it remains to show that $|P_{\text{gain}}^1(m) \cup P_{\text{gain}}^2(m)| = k$. To do this we exhibit a bijection $\Lambda^{-1}[0, m] \rightarrow P_{\text{gain}}^1(m) \cup P_{\text{gain}}^2(m)$. First, for $(j, l) \in \Lambda^{-1}[0, m]$ we map (j, l) to $l(q+1) + j + 1 \in P_{\text{gain}}^1(m)$ if $0 \leq j \leq q$ and $0 \leq l \leq \zeta_{\text{gain}} - 1$. Now we are left with defining a bijection F

$$\{(j, l) \in \Lambda^{-1}[0, m] : 0 \leq j \leq q, \zeta_{\text{gain}} \leq l \leq q-1 \text{ or } j \geq q+1\} \rightarrow P_{\text{gain}}^2(m) \text{ by}$$

$$F(j, l) = \begin{cases} \zeta_{\text{gain}}(q+1) + jq + (l - \zeta_{\text{gain}}) + 1 & \text{if } \zeta_{\text{gain}} \leq l \leq q-1, \\ \zeta_{\text{gain}}(q+1) + (j - q - 1)q + (l + \frac{q+q_2}{2}) + 1 & \text{if } 0 \leq l \leq \zeta_{\text{gain}} - 1. \end{cases}$$

It is easy to check that F maps into $P_{\text{gain}}^2(m)$ and F is one-to-one since for $\zeta_{\text{gain}} \leq l \leq q-1$ and $0 \leq l' \leq \zeta_{\text{gain}} - 1$, $0 \leq l - \zeta_{\text{gain}} \leq \frac{q+q_2}{2} - 1 < \frac{q+q_2}{2} \leq l' + \frac{q+q_2}{2} \leq q-1$. Finally we prove F is onto. For $i \in P_{\text{gain}}^2(m)$, such that $i = \zeta_{\text{gain}}(q+1) + jq + l + 1$ for $(j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q+1)]$, we put

$$(j', l') = \begin{cases} (j, l + \zeta_{\text{gain}}) & \text{if } 0 \leq l \leq \frac{q+q_2}{2} - 1, \\ (j + q + 1, l - \frac{q+q_2}{2}) & \text{if } \frac{q+q_2}{2} \leq l \leq q-1. \end{cases}$$

It is straightforward to see that (i) $(j', l') \in \Lambda^{-1}[0, m]$; (ii) if $j' \leq q$, then $\zeta_{\text{gain}} \leq l' \leq q-1$; and (iii) $F((j', l')) = i$. This completes the proof for $P_{\text{gain}}(m)$, and similarly for the points of fall. \square

EXAMPLE 4.5. If $q = 3$, then $P_{\text{gain}}(13) = [1, 9] \cup \{11, 14\}$ and $P_{\text{fall}}(13) = \{16\} \cup [18, 27]$, giving $s(C_{13}) = W(C_{13}) = \nabla^i(C_{\mathcal{L}}(D, 13Q_{\infty})) = 11$ using Theorem 3.9, but $s(C_{13}) = \nabla(C_{13}) + 1$.

Proof. The coordinate order of C_{13} is

$$\begin{aligned} & Q_{1,0,0} < Q_{1,1,0} < Q_{1,2,0} < Q_{1,3,0} < Q_{0,0,0} < Q_{0,0,1} < Q_{0,0,2} < Q_{2,0,0} < Q_{2,0,1} \\ & < Q_{2,0,2} < Q_{2,1,0} < Q_{2,1,1} < Q_{2,1,2} < Q_{2,2,0} < Q_{2,2,1} < Q_{2,2,2} < Q_{2,3,0} < Q_{2,3,1} \\ & < Q_{2,3,2} < Q_{1,0,1} < Q_{1,1,1} < Q_{1,2,1} < Q_{1,3,1} < Q_{1,0,2} < Q_{1,1,2} < Q_{1,2,2} < Q_{1,3,2}. \end{aligned}$$

We use the notation of the proof of Proposition 4.4. We note that $\zeta_{\text{gain}} = 1$. Thus for $0 \leq j \leq q$ and $0 \leq l \leq \zeta_{\text{gain}} - 1$, $jq + l(q+1) \leq 9 \leq 13$, so that $(j, l) \in \Lambda^{-1}[0, 13]$. Thus $P_{\text{gain}}^1(13)$ is the set of initial points of $z_{j,0}^{\text{gain}}$ for $0 \leq j \leq 3$, which are as follows.

(j, l)	z_{jl}^{gain}	Initial place	Initial point
$(0, 0)$	1	$Q_{1,0,0}$	1
$(1, 0)$	$(x - \alpha_{1,0})$	$Q_{1,1,0}$	2
$(2, 0)$	$(x - \alpha_{1,0})(x - \alpha_{1,1})$	$Q_{1,2,0}$	3
$(3, 0)$	$(x - \alpha_{1,0})(x - \alpha_{1,1})(x - \alpha_{1,2})$	$Q_{1,3,0}$	4

Thus $P_{\text{gain}}^1(13) = [1, 4]$. Next we consider $P_{\text{gain}}^2(13)$. Now we have

$$\alpha_0 = \alpha_{0,0}, \alpha_1 = \alpha_{2,0}, \alpha_2 = \alpha_{2,1}, \alpha_3 = \alpha_{2,2}, \alpha_4 = \alpha_{2,3}$$

so that $a(0) = 0$, $a(1) = a(2) = a(3) = a(4) = 2$. Then $P_{\text{gain}}^2(13)$ is the set of initial points of $z_{j,l+\zeta_{\text{gain}}}^{\text{gain}}$ such that $(j, l) \in \Lambda^{-1}[0, 13 - \zeta_{\text{gain}}(q+1)] = \Lambda^{-1}[0, 9]$ and

$$\Lambda^{-1}[0, 9] = \{(0, 0), (1, 0), (0, 1), (2, 0), (1, 1), (0, 2), (3, 0)\},$$

giving the following.

$(j, l+1)$	$z_{j,l+1}^{\text{gain}}$	Initial place	Initial point
(0, 1)	$(y - \beta_{1,0})$	$Q_{0,0,0}$	5
(1, 1)	$(y - \beta_{1,0})(x - \alpha_0)$	$Q_{2,0,0}$	8
(0, 2)	$(y - \beta_{1,0})(y - \beta_{0,0})$	$Q_{0,0,1}$	6
(2, 1)	$(y - \beta_{1,0})(x - \alpha_0)(x - \alpha_1)$	$Q_{2,1,0}$	11
(1, 2)	$(y - \beta_{1,0})(x - \alpha_0)(y - \beta_{2,0})$	$Q_{2,0,1}$	9
(0, 3)	$(y - \beta_{1,0})(y - \beta_{0,0})(y - \beta_{0,1})$	$Q_{0,0,2}$	7
(3, 1)	$(y - \beta_{1,0})(x - \alpha_0)(x - \alpha_1)(x - \alpha_2)$	$Q_{2,2,0}$	14

Thus $P_{\text{gain}}^2(13) = \{5, 8, 6, 11, 9, 7, 14\} = [5, 9] \cup \{11, 14\}$ and $P_{\text{gain}}(13) = [1, 9] \cup \{11, 14\}$, and similarly for $P_{\text{fall}}(13)$. We have $P_{\text{gain}}(13) < P_{\text{fall}}(13)$ and so $s(C_{13}) = 11$. \square

From Propositions 4.2 and 4.4 we have that, if (i) $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$ or (ii) $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$ or (iii) $\zeta_{\text{gain}} = \zeta_{\text{fall}}$ and $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$, then $P_{\text{fall}}(m) = n - P_{\text{gain}}(m) + 1$. In these cases the following useful property holds.

REMARK 4.6. *For a length n code C , if $P_{\text{fall}}(C) = n - P_{\text{gain}}(C) + 1$, then $s_{n-i}(C) = s_i(C)$ for $0 \leq i \leq n$. In particular, for $m \in J(n, g)$, if (i) q is odd and $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$ or $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$ or (ii) q is even and $0 \leq M^\circ \leq q$, then $s_i(C_m) = s_{n-i}(C_m)$ for $0 \leq i \leq n$. The same holds for $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ if m^\perp satisfies (i) or (ii).*

Proof. The proof is similar to that of [6, Proposition 2.5] and in fact can be modified to hold for branch complexity as in [6, Proposition 2.5]. We put $P_{\text{gain}}^{i,+}(C) = |P_{\text{gain}}(C) \cap [i+1, n]|$ and $P_{\text{fall}}^{i,+}(C) = |P_{\text{fall}}(C) \cap [i+1, n]|$. Of course, with $k = \dim(C)$,

$$P_{\text{gain}}^{i,+}(C) = k - P_{\text{gain}}^{i,-}(C) \text{ and } P_{\text{fall}}^{i,+}(C) = k - P_{\text{fall}}^{i,-}(C)$$

for any linear code C . The condition $P_{\text{fall}}(C) = n - P_{\text{gain}}(C) + 1$ also implies that

$$P_{\text{gain}}^{i,-}(C) = P_{\text{fall}}^{n-i,+}(C) \text{ and } P_{\text{fall}}^{i,-}(C) = P_{\text{gain}}^{i,+}(C).$$

Thus, from (9), we have

$$\begin{aligned} s_i(C) &= P_{\text{fall}}^{n-i,+}(C) - P_{\text{gain}}^{i,+}(C) \\ &= (k - P_{\text{fall}}^{n-i,-}(C)) - (k - P_{\text{gain}}^{n-i,-}(C)) = s_{n-i}(C). \quad \square \end{aligned}$$

REMARK 4.7. *If $C \in [C_{\mathcal{L}}(D, 14Q_\infty)]$ is ordered by O1, then as in the proof of Proposition 4.2 $P_{\text{gain}}(C) = [1, 11] \cup \{13\}$ and $P_{\text{fall}}(C) = \{15\} \cup [17, 27]$, so that $s(C) = 12$. However, if C is ordered by O2, $P_{\text{gain}}(14) = [1, 9] \cup [11, 12] \cup \{14\}$ and $P_{\text{fall}}(14) = \{13, 16\} \cup [18, 27]$, giving $s(C_{14}) = W(C_{14}) - 1 = \nabla(C_{14}) = 11$. Thus O2 is strictly better than O1 for $m=14$.*

If $C \in [C_{\mathcal{L}}(D, 15Q_\infty)]$ is ordered by O1, then as in the proof of Proposition 4.2, $P_{\text{gain}}(15) = [1, 11] \cup \{13, 16\}$ and $P_{\text{fall}}(15) = \{12, 15\} \cup [17, 27]$, giving $s(C_{15}) = \nabla(C_{15}) = W(C_{15}) - 2 = 11$. However, if C is ordered by O2, we get $P_{\text{gain}}(15) = [1, 12] \cup \{14\}$ and $P_{\text{fall}}(15) = \{13\} \cup [18, 27]$, giving $s(C_{15}) = 12$. Thus O1 is strictly better than O2 for $m = 15$.

To summarize, for $q = 2, 3$ and $m \in J(n, g) \subseteq I(n, g)$, $s(C_m) = \nabla^r(C_{\mathcal{L}}(D, mQ_\infty))$. Thus, in these cases $s(C_m) = s[C_{\mathcal{L}}(D, mQ_\infty)]$, and the coordinate order for C_m is optimal with regard to $s(C_m)$. In fact, except for $q = 3$ and $m \in \{11, 18\}$, $s(C_m) = \nabla(C_m) < W(C_m)$.

Another characterization of the points of gain and fall of C_m . We now characterize $P_{\text{gain}}(m)$ and $P_{\text{fall}}(m)$ as runs, i.e., as sequences of noncontiguous intervals of integers. This is useful since $s(C_m)$ must be attained at the end of a run of points of gain. Thus to determine $s(C_m)$, we need only to find the maximum of $s_i(C_m)$ over those i that end a run of points of gain, i.e. over those i such that $i \in P_{\text{gain}}(m)$ and $i + 1 \notin P_{\text{gain}}(m)$.

We begin by combining Propositions 4.2 and 4.4 for a common development of the cases (i) $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$ or $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$ and (ii) $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$. First, we extend the definitions of ζ_{gain} and ζ_{fall} as follows:

$$\zeta_{\text{gain}} = \begin{cases} 0 & \text{for } 0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}, \\ \frac{q-q_2}{2} & \text{for } \frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}, \\ q & \text{for } q - \frac{M^\bullet}{2} \leq M^\circ \leq q \end{cases}$$

and

$$\zeta_{\text{fall}} = \begin{cases} 0 & \text{for } 0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}, \\ \frac{q+q_2}{2} & \text{for } \frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}, \\ q & \text{for } q - \frac{M^\bullet}{2} \leq M^\circ \leq q. \end{cases}$$

PROPOSITION 4.8. *For $m \in J(n, g)$, $P_{\text{gain}}(m) = P_{\text{gain}}^1(m) \cup P_{\text{gain}}^2(m)$ and $P_{\text{fall}}(m) = P_{\text{fall}}^1(m) \cup P_{\text{fall}}^2(m)$, where*

$$\begin{aligned} P_{\text{gain}}^1(m) &= \{l(q+1) + j + 1 : (j, l) \in \Lambda^{-1}[0, m], 0 \leq j \leq q, 0 \leq l \leq \zeta_{\text{gain}} - 1\}, \\ P_{\text{gain}}^2(m) &= \{\zeta_{\text{gain}}(q+1) + jq + l + 1 : (j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q+1)]\}, \\ P_{\text{fall}}^1(m) &= \{n - l(q+1) - j : (j, l) \in \Lambda^{-1}[0, m], 0 \leq j \leq q, 0 \leq l \leq \zeta_{\text{fall}} - 1\}, \\ P_{\text{fall}}^2(m) &= \{n - \zeta_{\text{fall}}(q+1) - jq - l : (j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{fall}}(q+1)]\}. \end{aligned}$$

Proof. From the examples above and Remark 4.7, we can assume that $q \geq 4$. For $\frac{q-M^\bullet-2}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$, the result is just a restatement of Proposition 4.4. Also, for $0 \leq M^\circ \leq \frac{q-M^\bullet-1}{2}$, the result states that $P_{\text{gain}}(m) = P_{\text{gain}}^2(m) = \{jq + l + 1 : (j, l) \in \Lambda^{-1}[0, m]\}$ and $P_{\text{fall}}(m) = P_{\text{fall}}^2(m) = \{n - jq - l : (j, l) \in \Lambda^{-1}[0, m]\}$, in agreement with Proposition 4.2. Therefore we are reduced to m such that $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$ for which $\zeta_{\text{gain}} = \zeta_{\text{fall}} = q$. Rewriting $j'q + l' + 1$ as $q(q+1) + (j' - q - 1)q + l' + 1$ and $q(q+1) + jq + l + 1$ as $(j + q + 1)q + l + 1$, we see that $P_{\text{gain}}^1(m) = \{j'q + l' + 1 : (j', l') \in \Lambda^{-1}[0, m], 0 \leq j' \leq q\}$.

We claim that $P_{\text{gain}}^2(m) = \{j'q + l' + 1 : (j', l') \in \Lambda^{-1}[0, m], j' \geq q + 1\}$. First, if $0 \leq j \leq q$ and $0 \leq l \leq q - 1$, then $(j, l) \in \Lambda^{-1}[0, m]$ since $q \geq 4$. Thus we need to show that

$$\{j'q + l' + 1 : 0 \leq j' \leq q, 0 \leq l' \leq q - 1\} = \{l(q+1) + j + 1 : 0 \leq j \leq q, 0 \leq l \leq q - 1\}.$$

If k is in the left-hand side, $k = j'q + l' + 1$ for some $0 \leq j' \leq q$ and $0 \leq l' \leq q - 1$. Put

$$(j, l) = \begin{cases} (l' - j' + q + 1, j' - 1) & \text{if } 0 \leq l' < j', \\ (l' - j', j') & \text{if } j' \leq l' \leq q - 1. \end{cases}$$

In either case, $0 \leq j \leq q$, $0 \leq l \leq q - 1$ and $l(q+1) + j + 1 = j'q + l' + 1 = k$, so that k is in the right-hand side. The reverse inclusion is similar. The result now follows

from Proposition 4.2 since for $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$, $P_{\text{fall}}^1(m) = n - P_{\text{gain}}^1(m) + 1$ and $P_{\text{fall}}^2(m) = n - P_{\text{fall}}^2(m) + 1$. \square

LEMMA 4.9. *If $\theta_{\text{gain}} = M^\bullet + M^\circ - \zeta_{\text{gain}}$ and $\theta_{\text{fall}} = M^\bullet + M^\circ - \zeta_{\text{fall}}$, then $0 \leq \theta_{\text{gain}} \leq q - 2$ and $-1 \leq \theta_{\text{fall}} \leq q - 2$.*

Proof. The proof is straightforward using Lemma 3.7. \square

THEOREM 4.10. *For $m \in J(n, g)$,*

1. $P_{\text{gain}}(m)$ is the union of
 - (a) $[1, m - 2g - \theta_{\text{gain}}]$;
 - (b) $\{m - 2g - \theta_{\text{gain}} + eq + f + 1 : 0 \leq e \leq q - 2 - \theta_{\text{gain}}, 0 \leq f \leq q - 2 - e\}$;
and
 - (c) $\{m - 2g - \theta_{\text{gain}} + eq + f + 1 : q - 1 - \theta_{\text{gain}} \leq e \leq q - 1, 0 \leq f \leq q - 1 - e\}$;
and
2. $P_{\text{fall}}(m)$ is the union of
 - (a) $[n - m + 2g + \theta_{\text{fall}} + 1, n]$;
 - (b) $\{n - m + 2g + \theta_{\text{fall}} - eq - f : 0 \leq e \leq q - 2 - \theta_{\text{fall}}, 0 \leq f \leq q - 2 - e\}$; and
 - (c) $\{n - m + 2g + \theta_{\text{fall}} - eq - f : q - 1 - \theta_{\text{fall}} \leq e \leq q - 1, 0 \leq f \leq q - 1 - e\}$.

Proof. As in the proof of Proposition 4.8, we assume that $q \geq 4$. We will use the fact that

$$(10) \quad m - 2g - \theta_{\text{gain}} = \zeta_{\text{gain}}(q + 1) + \left(\frac{q^2 - q_2}{2} + M^\bullet - q + 1 - \zeta_{\text{gain}} \right) q.$$

For convenience we put $R_{\text{gain}}^1(m) = [1, m - 2g - \theta_{\text{gain}}]$, $R_{\text{gain}}^2(m) = \{m - 2g - \theta_{\text{gain}} + eq + f + 1 : 0 \leq e \leq q - 2 - \theta_{\text{gain}}, 0 \leq f \leq q - 2 - e\}$, and $R_{\text{gain}}^3(m) = \{m - 2g - \theta_{\text{gain}} + eq + f + 1 : q - 1 - \theta_{\text{gain}} \leq e \leq q - 1, 0 \leq f \leq q - 1 - e\}$.

We show that $R_{\text{gain}}^1(m) \subseteq P_{\text{gain}}(m)$ in two steps. First we note that $P_{\text{gain}}^1(m) = [1, \zeta_{\text{gain}}(q + 1)]$, since for $q \geq 4$, $0 \leq j \leq q$ and $0 \leq l \leq \zeta_{\text{gain}} - 1 \leq q - 1$, $\Lambda(j, l) \leq 2q^2 - 1 < \frac{n-1}{2} \leq m$. Next we show that $[\zeta_{\text{gain}}(q + 1) + 1, m - 2g - \theta_{\text{gain}}] \subseteq P_{\text{gain}}^2(m)$. Now from (10) we have that for $\zeta_{\text{gain}}(q + 1) + 1 \leq k \leq m - 2g - \theta_{\text{gain}}$,

$$k = \zeta_{\text{gain}}(q + 1) + jq + l + 1 \text{ for some } 0 \leq j \leq \left(\frac{q^2 - q_2}{2} + M^\bullet - q - \zeta_{\text{gain}} \right) \\ \text{and } 0 \leq l \leq q - 1.$$

Also, if $0 \leq j \leq \left(\frac{q^2 - q_2}{2} + M^\bullet - q - \zeta_{\text{gain}} \right)$ and $0 \leq l \leq q - 1$, then, again using (10),

$$\Lambda(j, l) \leq \left(\frac{q^2 - q_2}{2} + M^\bullet - q - \zeta_{\text{gain}} \right) q + (q - 1)(q + 1) = m - \theta_{\text{gain}} - \zeta_{\text{gain}}(q + 1) - 1,$$

so that $(j, l) \in \Lambda^{-1}[0, m - \zeta_{\text{gain}}(q + 1)]$. Thus $k \in P_{\text{gain}}^2(m)$. Next we show that $R_{\text{gain}}^2(m) \cup R_{\text{gain}}^3(m) \subseteq P_{\text{gain}}^2(m)$. Take $k = m - 2g - \theta_{\text{gain}} + eq + f + 1$. Then, from (10), $k = \zeta_{\text{gain}}(q + 1) + jq + l + 1$, where $j = \left(\frac{q^2 - q_2}{2} + M^\bullet - q + 1 - \zeta_{\text{gain}} + e \right)$ and $l = f$. Also, again using (10), $\Lambda(j, l) = m - 2g - \theta_{\text{gain}} - \zeta_{\text{gain}}(q + 1) + (e + f)q + f$. Thus $k \in P_{\text{gain}}^2(m)$ if $(e + f)q + f \leq 2g + \theta_{\text{gain}}$. If $0 \leq e \leq q - 2 - \theta_{\text{gain}}$ and $0 \leq f \leq q - 2 - e$, then $(e + f)q \leq (q - 2)q = 2g - q$ and $f \leq q - 2$, so that $R_{\text{gain}}^2(m) \subseteq P_{\text{gain}}^2(m)$. If $q - 1 - \theta_{\text{gain}} \leq e \leq q - 1$ and $0 \leq f \leq q - 1 - e$, then $(e + f)q \leq 2g$ and $f \leq \theta_{\text{gain}}$, so that $R_{\text{gain}}^3(m) \subseteq P_{\text{gain}}^2(m)$.

Thus $\bigcup_{i=1}^3 R_{\text{gain}}^i(m) \subseteq P_{\text{gain}}(m)$, and it suffices to show that $|\bigcup_{i=1}^3 R_{\text{gain}}^i(m)| = |P_{\text{gain}}|$. We recall that $|P_{\text{gain}}| = \dim(C_m)$ and since $2g - 2 < m < n$, $\dim(C_m) =$

$m - g + 1$. Also,

$$\left| \bigcup_{i=1}^3 R_{\text{gain}}^i(m) \right| = (m - 2g - \theta_{\text{gain}}) + \sum_{e=0}^{q-1} (q-1-e) + (\theta_{\text{gain}} + 1) = m - 2g + 1 + \sum_{e=0}^{q-1} e = m - g + 1.$$

The proof for $P_{\text{fall}}(m)$ is similar and we omit the details. \square

5. When the DLP bound is tight. Here we use Theorem 4.10 to determine $s(C_m)$. We know (from Corollary 3.10 and Proposition 3.12) that $s[C_{\mathcal{L}}(D, mQ_{\infty})] > \nabla(C_m)$ for just under half of the m in the range $I(n, g)$. We show that for the remaining m in this range, $s(C_m) = \nabla(C_m)$. As a consequence, we have determined $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ and a coordinate order that achieves $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ for such m . For those m with $s(C_m) > \nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$ we compare the upper bound, $s(C_m)$, on $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ with the lower bound $\nabla^t[C_{\mathcal{L}}(D, mQ_{\infty})]$ given in Corollary 3.10. When q is odd, these bounds meet for over three quarters of those m in $I(n, g)$, but when q is even, the bounds meet for only a little over one half of those m in $I(n, g)$.

Determining $s(C_m)$. As discussed in section 4, it suffices to find the maximum of $s_i(C_m)$ over those i such that $i \in P_{\text{gain}}(m)$ and $i + 1 \notin P_{\text{gain}}(m)$. From Theorem 4.10, there are only $q + 1$ such i . Thus concentrating on these i is significantly simpler. Therefore we calculate $s_i(C_m)$ for these $q + 1$ values of i (in Proposition 5.5) by determining $P_{\text{gain}}^{i,-}(m)$ and $P_{\text{fall}}^{i,-}(m)$ (in Lemmas 5.1 and 5.4). We determine which of these i gives the largest $s_i(C_m)$ (in Lemma 5.6). This enables us to write down $s(C_m)$ (in Theorem 5.7).

Early on we introduce a variable $\eta = \eta(m)$ which plays a crucial role in the proofs and statements of many of the results, and we end with a table of $s(C_m)$ for $m \in [\frac{n-1}{2}, \frac{n-3}{2} + g]$ when $q \in \{2, 3, 4, 5, 7, 8\}$.

We begin by determining $s(C_m)$ for $m \in J(n, g)$. We note first that $\theta_{\text{gain}} = M^{\bullet} + M^{\circ} - \zeta_{\text{gain}}$ and $\theta_{\text{fall}} = M^{\bullet} + M^{\circ} - \zeta_{\text{fall}}$, where ζ_{gain} and ζ_{fall} were defined just before Proposition 4.8.

As noted above, $s_i(C_m) = s(C_m)$ for some i such that $i \in P_{\text{gain}}(m)$ and $i + 1 \notin P_{\text{gain}}(m)$. From Theorem 4.10 such i are either (i) of the form $m - 2g - \theta_{\text{gain}} + eq + (q - 1 - e)$ for some $-1 \leq e \leq q - 2 - \theta_{\text{gain}}$ or (ii) of the form $m - 2g + \theta_{\text{gain}} + eq + (q - e)$ for some $q - 1 - \theta_{\text{gain}} \leq e \leq q - 1$. Thus putting

$$i_e = \begin{cases} m - 2g - \theta_{\text{gain}} + eq + (q - 1 - e) & \text{for } -1 \leq e \leq q - 2 - \theta_{\text{gain}}, \\ m - 2g - \theta_{\text{gain}} + eq + (q - e) & \text{for } q - 1 - \theta_{\text{gain}} \leq e \leq q - 1, \end{cases}$$

we have

$$(11) \quad s(C_m) = \max\{s_{i_e}(C_m) : -1 \leq e \leq q - 1\}.$$

From (9), $s_{i_e}(C_m) = P_{\text{gain}}^{i_e,-}(m) - P_{\text{gain}}^{i_e,+}(m)$, so we wish to determine $P_{\text{gain}}^{i_e,-}(m)$ and $P_{\text{fall}}^{i_e,-}(m)$ for $-1 \leq e \leq q - 1$. The first of these is straightforward.

LEMMA 5.1. For $m \in J(n, g)$,

$$P_{\text{gain}}^{i_e,-}(m) = \begin{cases} k - \binom{q-e}{2} + (q - 2 - \theta_{\text{gain}} - e) & \text{for } -1 \leq e \leq q - 2 - \theta_{\text{gain}}, \\ k - \binom{q+e}{2} & \text{for } q - 1 - \theta_{\text{gain}} \leq e \leq q - 1. \end{cases}$$

Proof. Since $2g - 2 < m < n$ we have $k = m - g + 1$. For $-1 \leq e \leq q - 2 - \theta_{\text{gain}}$, Theorem 4.10 gives

$$P_{\text{gain}}^{i_e,-}(m) = m - 2g - \theta_{\text{gain}} + \sum_{\nu=0}^e (q - 1 - \nu) = k - g - (\theta_{\text{gain}} + 1) + \sum_{\nu=q-1-e}^{q-1} \nu.$$

The first case follows since $\sum_{\nu=q-1-e}^{q-1} \nu = g - \binom{q-1-e}{2}$ and $\binom{q-1-e}{2} = \binom{q-e}{2} - (q-1-e)$. In the second case,

$$P_{\text{gain}}^{i_e, -}(m) = m - g + 1 - \binom{q-e}{2} - (q-2 - \theta_{\text{gain}} - e) - (e - (q-2 - \theta_{\text{gain}})). \quad \square$$

For $P_{\text{fall}}^{i_e, -}(m)$ it is convenient to introduce some more notation. For fixed m we put

$$\zeta_{\text{norm}} = \frac{\zeta_{\text{gain}} + \zeta_{\text{fall}}}{q}.$$

Thus ζ_{norm} is 0, 1, or 2, depending on whether $0 \leq M^\circ \leq \frac{q-M^\bullet-2}{2}$, $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq q - \frac{M^\bullet+1}{2}$, or $q - \frac{M^\bullet}{2} \leq M^\circ \leq q$. Also, we put

$$\eta = 2q - 2M^\bullet + q_2 - \zeta_{\text{norm}} - 3.$$

In Lemma 5.4 and Proposition 5.5 we will see a symmetry between the roles of e in $P_{\text{gain}}^{i_e, -}(m)$ and $\eta - e$ in $P_{\text{fall}}^{i_e, -}(m)$. We will see in Lemma 5.6 that $s_{i_e}(C_m)$ is maximized near $\frac{\eta}{2}$, and hence η appears naturally in our formula for $s(C_m)$.

LEMMA 5.2. $q-1 \leq \eta \leq 2q-3$.

Proof. First, it follows from Lemma 3.7 that

$$(12) \quad \eta \geq \left\{ \begin{array}{ll} 2q - (q-3) + 1 - 2 - 3 = q-1 & \text{if } q \text{ is odd,} \\ 2q - (q-4) - 2 - 3 = q-1 & \text{if } q \text{ is even and } M^\circ > 0, \\ 2q - (q-2) - 3 = q-1 & \text{if } q \text{ is even and } M^\circ = 0 \end{array} \right\} = q-1.$$

Next, clearly $\eta \leq 2q-2$, with equality only if $M^\bullet = \zeta_{\text{norm}} = 0$ and $q_2 = 1$. However, from Lemma 3.7, if $M^\bullet = 0$ and q is odd, then $M^\circ \geq \frac{q-1}{2}$ so that $\zeta_{\text{norm}} \geq 1$. \square

Now, in order to use Theorem 4.10 to calculate $P_{\text{fall}}^{i_e, -}(m)$, we need to write i_e as $n-m+2g+\theta_{\text{fall}}-e'q-f$ for some, preferably nonnegative, integer e' and $0 \leq f \leq q-1$. We could then determine an expression for $P_{\text{fall}}^{i_e, -}(m)$ in terms of e' and f in a similar way to the proof of Lemma 5.1, except that f would add complications. This would give us an expression for $s_{i_e}(C_m)$ in terms of e , e' , and f . To maximize this over $-1 \leq e \leq q-1$ we would need to relate e' and f to e . Fortunately, these relationships are reasonably simple.

LEMMA 5.3. *Let $m \in I(n, g)$ and $-1 \leq e \leq q-1$. If we write*

$$i_e = n - m + 2g + \theta_{\text{fall}} - e'q - f \text{ for some } 0 \leq f \leq q-1,$$

then $e' = \eta - e$ and

$$f = \begin{cases} e+1 & \text{for } -1 \leq e \leq q-2 - \theta_{\text{gain}}, \\ e & \text{for } q-1 - \theta_{\text{gain}} \leq e \leq q-1. \end{cases}$$

In particular, $e' \geq 0$. Also, if $e \leq \eta - q + 1 + \theta_{\text{fall}}$, then $q - \eta + e - f \leq 0$.

Proof. For $-1 \leq e \leq q-2 - \theta_{\text{gain}}$, we have $(e+e')q + (q-1-e+f) = n-2m+4g+\theta_{\text{gain}}+\theta_{\text{fall}}$. Now $2m = n - q_2q + 2M^\bullet(q+1) + 2M^\circ$, $4g = 2q^2 - 2q$, and $\theta_{\text{gain}} + \theta_{\text{fall}} = 2M^\bullet + 2M^\circ - \zeta_{\text{norm}}q$, giving $(e+e')q + (q-1-e+f) = (2q-2M^\bullet+q_2-\zeta_{\text{norm}})q$ which implies that $f = e+1$ (since $q-1-e > 0$ from Lemma 4.9) and $e' = \eta - e$. Similarly, for $q-1 - \theta_{\text{gain}} \leq e \leq q-1$ we get $(e+e')q + (q-e+f) = (\eta+1)q$, giving $f = e$ and $e' = \eta - e$.

For the second part we have $\eta \geq q - 1$ (from Lemma 5.2) and $f \geq e$ (from the first part). Thus $q - \eta + e - f \leq 1$ with equality only if $\eta = q - 1$ and $f = e$. We show that, for $e \leq \eta - q + 1 + \theta_{\text{fall}}$, it is not possible that $\eta = q - 1$ and $f = e$. First, $f = e$ implies that $e \geq q - 1 - \theta_{\text{gain}}$. Also, $\eta = q - 1$ and $e \leq \eta - q + 1 + \theta_{\text{fall}}$ imply that $e \leq \theta_{\text{fall}}$. Thus $q - 1 - \theta_{\text{gain}} \leq e \leq \theta_{\text{fall}}$ so that, adding θ_{gain} to both sides,

$$(13) \quad 2M^\bullet + 2M^\circ - q\zeta_{\text{norm}} \geq q - 1.$$

Now, as in (12), $\eta = q - 1$ implies that either (i) $\zeta_{\text{norm}} = 2$ and $M^\bullet \leq \frac{q-3}{2}$ or (ii) $M^\bullet = \frac{q-2}{2}$ and $M^\circ = 0$. Each of these clearly contradicts (13). \square

LEMMA 5.4. For $m \in J(n, g)$,

$$P_{\text{fall}}^{i_e, -}(m) = \begin{cases} \binom{q-\eta+e}{2} & \text{for } -1 \leq e \leq \eta - q + 1 + \theta_{\text{fall}}, \\ \binom{q-\eta+e}{2} - (q - 2 - \theta_{\text{fall}} - \eta + e) & \text{for } \eta - q + 2 + \theta_{\text{fall}} \leq e \leq q - 1. \end{cases}$$

Proof. We write $i_e = n - m + 2g + \theta_{\text{fall}} - e'q - f$ if $0 \leq f \leq q - 1$, as in Lemma 5.3, and work from Theorem 4.10. First, if $e' \geq q$, i.e., if $e \leq \eta - q$, then $P_{\text{fall}}^{i_e, -}(m) = 0$. We note also that, for $e \leq \eta - q$, $\binom{q-\eta+e}{2} = 0$. Next, if $q - 1 - \theta_{\text{fall}} \leq e' \leq q - 1$, i.e., if $\eta - q + 1 \leq e \leq \eta - q + 1 + \theta_{\text{fall}}$, then

$$\begin{aligned} P_{\text{fall}}^{i_e, -}(m) &= \sum_{\nu=1}^{q-1-e'} \nu + \max\{0, q - e' - f\} \\ &= \binom{q-\eta+e}{2} + \max\{0, q - \eta + e - f\} = \binom{q-\eta+e}{2}, \end{aligned}$$

the last equality following from the second part of Lemma 5.3. Finally (since $e' \geq 0$ by Lemma 5.3), if $0 \leq e' \leq q - 2 - \theta_{\text{fall}}$, i.e., if $\eta - q + 2 + \theta_{\text{fall}} \leq e \leq q - 1$ (since $\eta \geq q - 1$), then

$$\begin{aligned} P_{\text{fall}}^{i_e, -}(m) &= \sum_{\nu=1}^{q-1-e'} \nu - (q - 2 - \theta_{\text{fall}} - e') + \max\{0, q - e' - 1 - f\} \\ &= \binom{q-\eta+e}{2} - (q - 2 - \theta_{\text{fall}} - \eta + e) + \max\{0, q - \eta + e - f - 1\} \end{aligned}$$

and $q - \eta + e - f - 1 \leq 0$ since $\eta \geq q - 1$ and $f \geq e$, by Lemma 5.3. \square

We use the convention that, for $b \geq 0$, $\binom{a}{b} = 0$ if $a < b$. In particular,

$$\binom{a}{1} = \begin{cases} a & \text{for } a \geq 0, \\ 0 & \text{for } a \leq 0, \end{cases} \quad \binom{a}{0} = \begin{cases} 1 & \text{for } a \geq 0, \\ 0 & \text{for } a < 0, \end{cases} \quad \binom{a}{b} - \binom{a-1}{b} = \binom{a-1}{b-1},$$

where $b \geq 1$. Lemmas 5.1 and 5.4, together with (9), give the following proposition.

PROPOSITION 5.5. For $m \in J(n, g)$,

$$s_{i_e}(C_m) = k - \binom{q-e}{2} - \binom{q-\eta+e}{2} + \binom{q-2-\theta_{\text{gain}}-e}{1} + \binom{q-2-\theta_{\text{fall}}-\eta+e}{1}.$$

Now we determine for which e , $-1 \leq e \leq q - 1$, $s_{i_e}(C_m)$ is maximized.

LEMMA 5.6. For $m \in J(n, g)$, $s_{i_e}(C_m)$ is maximized

1. at $e = \lfloor \frac{\eta}{2} \rfloor$ if $\eta \leq 2q - 6 - 2\theta_{\text{fall}}$ or
2. at $e = \lceil \frac{\eta}{2} \rceil$ if $\eta \geq 2q - 5 - 2\theta_{\text{fall}}$.

Proof. From Proposition 5.5, with

$$\sigma(e) = \binom{q-e}{2} + \binom{q-\eta+e}{2} - \binom{q-2-\theta_{\text{gain}}-e}{1} - \binom{q-2-\theta_{\text{fall}}-\eta+e}{1},$$

we have $s_{i_e}(C_m) = k - \sigma(e)$ and maximizing $s_{i_e}(C_m)$ is equivalent to minimizing $\sigma(e)$ over $-1 \leq e \leq q-1$. Now, for $0 \leq e \leq q-1$,

$$\begin{aligned} \sigma(e) - \sigma(e-1) &= - \binom{q-e}{1} + \binom{q-\eta+e-1}{1} + \binom{q-2-\theta_{\text{gain}}-e}{0} \\ &\quad - \binom{q-3-\theta_{\text{fall}}-\eta+e}{0}. \end{aligned}$$

Thus, since $0 \leq \binom{q-2-\theta_{\text{gain}}-e}{0} \leq 1$, we have

$$(14) \quad \begin{aligned} e-q &\leq \sigma(e) - \sigma(e-1) \leq e-q+1 && \text{for } 0 \leq e \leq \eta-q+1, \\ 2e-\eta-1 &\leq \sigma(e) - \sigma(e-1) \leq 2e-\eta && \text{for } \eta-q+2 \leq e \leq \eta-q+2+\theta_{\text{fall}}, \\ 2e-\eta-2 &\leq \sigma(e) - \sigma(e-1) \leq 2e-\eta-1 && \text{for } \eta-q+3+\theta_{\text{fall}} \leq e \leq q-1. \end{aligned}$$

First, for $0 \leq e \leq \eta-q+1$, (14) implies that $\sigma(e) - \sigma(e-1) \leq \eta-2q+2 \leq 0$, so that $\sigma(e)$ is minimized over $-1 \leq e \leq \eta-q+1$ at $e = \eta-q+1$. Thus it is sufficient to determine where $\sigma(e)$ is minimized over $\eta-q+1 \leq e \leq q-1$. We note that, since $\eta \leq 2q-3$ (Lemma 5.2),

$$\eta-q+1 \leq \left\lfloor \frac{\eta}{2} \right\rfloor \leq \left\lceil \frac{\eta}{2} \right\rceil \leq q-1.$$

Now, for $\eta-q+2 \leq e \leq \eta-q+2+\theta_{\text{fall}}$, (14) implies that if $e \leq \lfloor \frac{\eta}{2} \rfloor$, then $\sigma(e) - \sigma(e-1) \leq 0$ and if $e \geq \lfloor \frac{\eta}{2} \rfloor + 1 \geq \frac{\eta+1}{2}$, then $\sigma(e) - \sigma(e-1) \geq 0$. Similarly, for $\eta-q+3+\theta_{\text{fall}} \leq e \leq q-1$, (14) implies that if $e \leq \lfloor \frac{\eta+1}{2} \rfloor = \lceil \frac{\eta}{2} \rceil$, then $\sigma(e) \leq \sigma(e-1)$ and if $e \geq \lceil \frac{\eta}{2} \rceil + 1$, then $\sigma(e) \geq \sigma(e-1)$. Thus

1. if $\lceil \frac{\eta}{2} \rceil \leq \eta-q+2+\theta_{\text{fall}}$, then $\sigma(e)$ is minimized over $\eta-q+1 \leq e \leq q-1$ at $e = \lceil \frac{\eta}{2} \rceil$ and
2. if $\lceil \frac{\eta}{2} \rceil \geq \eta-q+3+\theta_{\text{fall}}$, then $\sigma(e)$ is minimized over $\eta-q+1 \leq e \leq q-1$ at $e = \lceil \frac{\eta}{2} \rceil$.

This leaves the case $\lfloor \frac{\eta}{2} \rfloor = \lceil \frac{\eta}{2} \rceil - 1 = \eta-q+2+\theta_{\text{fall}}$, i.e., $\eta = 2q-5-2\theta_{\text{fall}}$. In this case, the above analysis implies that $\sigma(e)$ is minimized at either $\lfloor \frac{\eta}{2} \rfloor = \eta-q+2+\theta_{\text{fall}} = q-3-\theta_{\text{fall}}$ or $\lceil \frac{\eta}{2} \rceil = \eta-q+3+\theta_{\text{fall}} = q-2-\theta_{\text{fall}}$. Also, we have

$$\sigma(q-2-\theta_{\text{fall}}) - \sigma(q-3-\theta_{\text{fall}}) = -(\theta_{\text{fall}}+2) + (\theta_{\text{fall}}+2) + \binom{\theta_{\text{fall}}-\theta_{\text{gain}}}{0} - 1 \leq 0,$$

so that $\sigma(e)$ is minimized at $\lceil \frac{\eta}{2} \rceil$.

Finally, we note that if $\eta \geq 2q-3-2\theta_{\text{fall}}$, then $-\eta \leq -2q+3+2\theta_{\text{fall}}$, so that adding $2\eta+1$ to both sides and dividing by 2 implies

$$\left\lceil \frac{\eta}{2} \right\rceil \leq \frac{\eta+1}{2} \leq \eta-q+2+\theta_{\text{fall}},$$

and we are in case 1 above. Also, if $\eta = 2q-4-2\theta_{\text{fall}}$ we have $\lceil \frac{\eta}{2} \rceil = \eta-q+2+\theta_{\text{fall}}$, and again we are in case 1. Similarly for $\eta \leq 2q-6-2\theta_{\text{fall}}$ we are in case 2 above. \square

Proposition 5.5 and Lemma 5.6 give us the following theorem.

THEOREM 5.7. *For $m \in J(n, g)$,*

$$s(C_m) = \begin{cases} k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta}{2} \rceil}{2} + (2q - 4 - \theta_{\text{fall}} - \theta_{\text{gain}} - \eta) & \text{for } \eta \leq 2q - 6 - 2\theta_{\text{fall}}, \\ k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta}{2} \rceil}{2} + 1 & \text{for } \eta = 2q - 5 - 2\theta_{\text{fall}}, \\ k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta}{2} \rceil}{2} & \text{for } \eta \geq 2q - 4 - 2\theta_{\text{fall}}. \end{cases}$$

Proof. The result follows since

1. for $\eta \leq 2q - 6 - 2\theta_{\text{fall}}$, $q - 2 - \theta_{\text{gain}} - \lfloor \frac{\eta}{2} \rfloor \geq 0$ and $q - 2 - \theta_{\text{fall}} - \lceil \frac{\eta}{2} \rceil \geq 1$;
2. for $\eta = 2q - 5 - 2\theta_{\text{fall}}$, $q - 2 - \theta_{\text{gain}} - \lceil \frac{\eta}{2} \rceil \leq 0$ and $q - 2 - \theta_{\text{fall}} - \lfloor \frac{\eta}{2} \rfloor = 1$; and
3. for $\eta \geq 2q - 4 - 2\theta_{\text{gain}}$, $q - 2 - \theta_{\text{gain}} - \lceil \frac{\eta}{2} \rceil \leq 0$ and $q - 2 - \theta_{\text{fall}} - \lfloor \frac{\eta}{2} \rfloor \leq 0$.

For example, $\eta \leq 2q - 6 - 2\theta_{\text{fall}}$ implies that $\lfloor \frac{\eta}{2} \rfloor \leq q - 3 - \theta_{\text{fall}}$, so that

$$q - 2 - \theta_{\text{gain}} - \left\lfloor \frac{\eta}{2} \right\rfloor \geq 1 + \theta_{\text{fall}} - \theta_{\text{gain}} = 1 + \zeta_{\text{gain}} - \zeta_{\text{fall}} \geq 0.$$

The other equalities and inequalities follow similarly. \square

Of course, Theorem 5.7 essentially gives the values of $s(C_m)$ for $I(n, g)$ since $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ implies $m^\perp \in J(n, g)$ and $s(C_m) = s(C_{m^\perp})$.

Table 3 gives $s(C_m)$ for $q \in \{2, 3, 4, 5, 7, 8\}$ and $m \in J(n, g)$. Comparing these values of $s(C_m)$ with the values of $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ given in Table 2 we have $s(C_m) = \nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ except for $q = 5$ and $m = 70$, $q = 7$ and $m \in \{182, 189, 190\}$, and $q = 8$ and $m \in \{268, 272, 276, 277, 280, 281\}$. In particular, $s(C_m)$ achieves the DLP bound for C_m for $q \in \{2, 3, 4, 5, 7, 8\}$ and $m \in I(n, g)$ when this is not excluded by Corollary 3.10, i.e., whenever the entry for m or m^\perp in Table 2 is not in boldface.

TABLE 3
 $s(C_m)$ for $q \in \{2, 3, 4, 5, 7, 8\}$ and $m \in J(n, g)$.

2	m	4													
	$s(C_m)$	3													
3	m	13	14	15											
	$s(C_m)$	11	11	11											
4	m	32	33	34	35	36	37								
	$s(C_m)$	26	27	27	28	28	28								
5	m	62	63	64	65	66	67	68	69	70	71				
	$s(C_m)$	53	53	54	54	55	56	56	56	57	56				
7	m	171	172	173	174	175	176	177	178	179	180				
	$s(C_m)$	151	151	152	153	153	154	155	156	156	156				
	m	181	182	183	184	185	186	187	188	189	190	191			
	$s(C_m)$	157	158	157	158	159	159	159	159	160	160	159			
8	m	256	257	258	259	260	261	262	263	264	265	266	267	268	269
	$s(C_m)$	228	229	230	231	231	232	233	234	234	234	235	236	237	236
	m	270	271	272	273	274	275	276	277	278	279	280	281	282	283
	$s(C_m)$	237	238	239	238	238	239	240	240	239	240	241	241	240	240

Comparing $s(C_m)$ with $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$. We start by reinterpreting $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$ in terms of η in Theorem 5.8. We use this to calculate (in Proposition 5.9) and hence to show (in Corollary 5.10) that $s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$ whenever this is not excluded by Corollary 3.10. This means that $s(C_m)$ achieves the DLP bound for C_m for just over half of those m in the range $[\frac{n-1}{2}, \frac{n-3}{2} + 2g]$. We then compare $s(C_m)$ with $\nabla^i(C_{\mathcal{L}}(D, mQ_\infty))$ in Table 4 and see that $s(C_m)$ achieves the bound

$\nabla^i(C_{\mathcal{L}}(D, mQ_{\infty}))$ for approximately a further quarter of those m in $[\frac{n-1}{2}, \frac{n-3}{3} + 2g]$ if q is odd but only for about a further $1/q$ of those m in $[\frac{n-1}{2}, \frac{n-3}{3} + 2g]$ if q is even.

Previously, we partitioned $J(n, g)$ into three subintervals, according to whether $0 \leq M^{\circ} \leq \frac{q-M^{\bullet}-2}{2}$, $\frac{q-M^{\bullet}-1}{2} \leq M^{\circ} \leq q - \frac{M^{\bullet}+1}{2}$, or $q - \frac{M^{\bullet}}{2} \leq M^{\circ} \leq q$. Now we consider a finer partition and say that $m \in J(n, g)$ satisfies (A), (B), (C), (D), or (E) according to whether (A) $0 \leq M^{\circ} \leq \frac{q-2}{2} - M^{\bullet}$, (B) $\frac{q-1}{2} - M^{\bullet} \leq M^{\circ} \leq \frac{q-M^{\bullet}-2}{2}$, (C) $\frac{q-M^{\bullet}-1}{2} \leq M^{\circ} \leq q - M^{\bullet} - 1$, (D) $q - M^{\bullet} \leq M^{\circ} \leq q - \frac{M^{\bullet}+1}{2}$, or (E) $q - \frac{M^{\bullet}}{2} \leq M^{\circ} \leq q$.

We compare $s(C_m)$ with $\nabla^i(C_{\mathcal{L}}(D, mQ_{\infty}))$ by reinterpreting Theorems 3.9 and 5.7 using (A)–(E).

THEOREM 5.8. *If $m \in J(n, g)$, then*

$$\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = \begin{cases} k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta+1}{2} \rceil}{2} & \text{if } m \text{ satisfies (A), (C), (E),} \\ k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta+1}{2} \rceil}{2} - (\theta_{\text{fall}} + \theta_{\text{gain}} - q + 2) & \text{otherwise.} \end{cases}$$

Proof. Take u and v as in the statement of Theorem 2.1. It is straightforward to show, using the characterization of (u, v) given in the proof of Lemma 3.8, that if m satisfies (A), (C), or (E), then $\eta = u - 1$ and $v = q - \theta_{\text{gain}} - \theta_{\text{fall}} - 2$ and if m satisfies (B) or (D), then $\eta = u$ and $v = 2q - \theta_{\text{gain}} - \theta_{\text{fall}} - 2$. Thus Theorem 2.1 implies that, for m satisfying (A), (C), or (E),

$$\begin{aligned} \nabla(C_{\mathcal{L}}(D, mQ_{\infty})) &= k - \binom{q - \lfloor \frac{\eta+1}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta+1}{2} \rceil}{2} \\ &\quad - \min \left\{ q - \left\lceil \frac{\eta+1}{2} \right\rceil, \theta_{\text{gain}} + \theta_{\text{fall}} + 2 \right\} \end{aligned}$$

and for m satisfying (B) or (D),

$$\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta}{2} \rceil}{2} - \min \left\{ q - \left\lceil \frac{\eta}{2} \right\rceil, \theta_{\text{gain}} + \theta_{\text{fall}} + 2 - q \right\}.$$

First, for m satisfying (A), (C), or (E) we have (i) $\lceil \frac{\eta+1}{2} \rceil \geq q - M^{\bullet} - 1$ if $\zeta_{\text{norm}} \in \{0, 1\}$ or (ii) $\lceil \frac{\eta+1}{2} \rceil \geq q - M^{\bullet} - 2$ if $\zeta_{\text{norm}} = 2$. Also, $\theta_{\text{gain}} + \theta_{\text{fall}} + 2 = 2M^{\bullet} + 2M^{\circ} - \zeta_{\text{gain}} - \zeta_{\text{fall}} + 2$ and (i) for $\zeta_{\text{norm}} = 0$, $2M^{\circ} - \zeta_{\text{gain}} - \zeta_{\text{fall}} \geq 0$, (ii) for $\zeta_{\text{norm}} = 1$, $2M^{\circ} - \zeta_{\text{gain}} - \zeta_{\text{fall}} \geq (q - M^{\bullet} - 1) - q = -M^{\bullet} - 1$ or (iii) for $\zeta_{\text{norm}} = 2$, $2M^{\circ} - \zeta_{\text{gain}} - \zeta_{\text{fall}} \geq (2q - M^{\bullet}) - 2q = M^{\bullet}$. Thus, for m satisfying (A), (C), or (E), $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$ is equal to

$$k - \binom{q - \lfloor \frac{\eta+1}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta+1}{2} \rceil}{2} - q - \left\lceil \frac{\eta+1}{2} \right\rceil = k - \binom{q - \lfloor \frac{\eta+1}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta+1}{2} \rceil + 1}{2}$$

as required. Similarly, for m satisfying (B) or (D) (so that $\zeta_{\text{norm}} \leq 1$) it is easy to see that $q - \lceil \frac{\eta}{2} \rceil \geq M^{\bullet} + 1$ and (by considering the cases that $\zeta_{\text{norm}} = 0$ and $\zeta_{\text{norm}} = 1$ separately) $\theta_{\text{gain}} + \theta_{\text{fall}} + 2 - q \leq M^{\bullet} + 1$. Thus, for m satisfying (B) or (D),

$$\nabla(C_{\mathcal{L}}(D, mQ_{\infty})) = k - \binom{q - \lfloor \frac{\eta}{2} \rfloor}{2} - \binom{q - \lceil \frac{\eta}{2} \rceil}{2} - (\theta_{\text{gain}} + \theta_{\text{fall}} + 2 - q)$$

as required. \square

Before comparing $s(C_m)$ with $\nabla^i(C_{\mathcal{L}}(D, mQ_{\infty}))$, we compare it with $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$. To do this we refine (A)–(E) as follows: if m satisfies (C), then we

say that m satisfies (C1), (C2), or (C3) if (C1) $\frac{q-M^\bullet-1}{2} \leq M^\circ \leq \frac{q-2}{2}$, (C2) $M^\circ = \frac{q-1}{2}$, or (C3) $\frac{q}{2} \leq M^\circ \leq q - M^\bullet - 1$.

PROPOSITION 5.9. For $m \in J(n, g)$,

$$s(C_m) - \nabla(C_{\mathcal{L}}(D, mQ_\infty)) = \begin{cases} 0 & \text{if } m \text{ satisfies (A),} \\ 2M^\bullet + 2M^\circ - q + 2 & \text{if } m \text{ satisfies (B),} \\ q - 2M^\circ - q_2 & \text{if } m \text{ satisfies (C1),} \\ 1 & \text{if } m \text{ satisfies (C2),} \\ 0 & \text{if } m \text{ satisfies (C3),} \\ 2M^\bullet + 2M^\circ - 2q + 2 & \text{if } m \text{ satisfies (D),} \\ 2q - 2M^\circ + 1 - q_2 & \text{if } m \text{ satisfies (E).} \end{cases}$$

Proof. Using $\eta = 2q - 2\theta_{\text{fall}} + 2M^\circ - 2\zeta_{\text{fall}} + q_2 - \zeta_{\text{norm}} - 3$, it is straightforward to see that if $M^\circ \leq q - 1$ and

1. if m satisfies (A), (B), (D), or (C3), then $\eta \geq 2q - 2\theta_{\text{fall}} - 4$;
2. if m satisfies (C1) or (E), then $\eta \leq 2q - 6 - \theta_{\text{fall}}$; or
3. if m satisfies (C2), then $\eta = 2q - 5 - \theta_{\text{fall}}$.

Also, if $m = q$ is odd, then $\eta = 2q - 2\theta_{\text{fall}} - 4$. Likewise, if $m = q$ is even, then $\eta = 2q - 2\theta_{\text{fall}} - 5$. The result then follows from Theorems 5.7 and 5.8 noting that, for cases (B) and (D), $\theta_{\text{gain}} + \theta_{\text{fall}} - q + 2 = 2M^\bullet + 2M^\circ - (\zeta_{\text{norm}} + 1)q + 2$ and for cases (C1) and (E) with $M^\circ \leq q - 1$, $2q - 4 - \theta_{\text{fall}} - \theta_{\text{gain}} - \eta = \zeta_{\text{norm}}q - 2M^\circ - q_2 + (\zeta_{\text{norm}} - 1)$. \square

It follows from Proposition 5.9 that $s(C_m)$ achieves the DLP bound for C_m as often as this is possible. We state this as the following corollary.

COROLLARY 5.10. For $m \in I(n, g)$, $s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$ if and only if $\Delta(m) = 0$.

Proof. Since for $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$, $\Delta(m) = \Delta(m^\perp)$, $\nabla(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla(C_{\mathcal{L}}(D, m^\perp Q_\infty))$, and $s(C_m) = s(C_{m^\perp})$, it suffices to show the result for $m \in J(n, g)$. It follows from the definition of $\Delta(m)$ for such m that $\Delta(m) = 0$ if and only if (i) m satisfies (A) or (ii) m satisfies (C3) or (iii) $q_2 = 1$ and $M^\circ = q$. These are exactly the values of M° for which Proposition 5.9 gives $s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_\infty))$. \square

EXAMPLE 5.11. If C_m is self-dual, then $\nabla(C_m) = s(C_m) = \frac{n}{2} - \frac{q^2}{4}$, where C_m has the lexicographic coordinate order. In particular, $s[C_m] = \frac{n}{2} - \frac{q^2}{4}$.

Proof. We know that q is a power of 2, $k = \frac{n}{2}$, and $m = \frac{n}{2} + g - 1 \in J(n, g) \subseteq I(n, g)$. From the definitions, $M^\bullet = \frac{q-2}{2}$ and $M^\circ = \zeta_{\text{norm}} = 0$. Also, $\nabla(C_m) = \frac{n}{2} - \frac{q^2}{4}$ by Theorem 5.8. The result now follows since $\Delta(m) = 0$. \square

We remark that the main result of [13] is Example 5.11 with $q \geq 4$. Corollary 5.10 and Proposition 3.12 imply that $\nabla(C_m)$ is attained for just over half the $m \in I(n, g)$. Explicitly, the proportion of these m for which the DLP bound is attained is $\frac{1}{2} + \frac{1}{2q}$ for q odd and $\frac{1}{2} + \frac{3q-5}{2(q^2-q-1)}$ for q even. Of course Corollary 5.10 implies that if m satisfies (A), (C3) or $M^\circ = q$ is odd, then

$$s[C_{\mathcal{L}}(D, mQ_\infty)] = \nabla(C_{\mathcal{L}}(D, mQ_\infty)) = s(C_m).$$

The increments on $s[C_{\mathcal{L}}(D, mQ_\infty)]$ given by Theorem 3.9 and Proposition 5.9 for all m in $J(n, g)$ (and hence implicitly also for $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$) are given in Table 4. The first entry is $\Delta(m)$ and the second is $s(C_m) - \nabla(C_{\mathcal{L}}(D, mQ_\infty))$. Thus our lower bound for $s[C_{\mathcal{L}}(D, mQ_\infty)]$ is $\nabla(C_{\mathcal{L}}(D, mQ_\infty)) = \nabla(C_{\mathcal{L}}(D, mQ_\infty)) + \Delta(m)$ and our upper bound for $s[C_{\mathcal{L}}(D, mQ_\infty)]$ is $\nabla(C_{\mathcal{L}}(D, mQ_\infty))$ plus the second entry,

TABLE 4
Table of bounds on $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ for $m \in J(n, g)$.

m satisfies	$\Delta(m)$	$s(C_m) - \nabla$	Range
(A)	0	0	0
(B)	$M^{\bullet} + M^{\circ} + 1 - \frac{q-q_2}{2}$	$2M^{\bullet} + 2M^{\circ} + 2 - q$	$M^{\bullet} + M^{\circ} + 1 - \frac{q+q_2}{2}$
(C1)	$\frac{q+q_2}{2} - M^{\circ}$	$q - q_2 - 2M^{\circ}$	$\frac{q-q_2}{2} - M^{\circ}$
(C2)	1	1	0
(C3)	0	0	0
(D)	$M^{\bullet} + M^{\circ} + 1 - q$	$2M^{\bullet} + 2M^{\circ} + 2 - 2q$	$M^{\bullet} + M^{\circ} + 1 - q$
(E)	$q - M^{\circ} + 1 - q_2$	$2q - 2M^{\circ} + 1 - q_2$	$q - M^{\circ}$

i.e., $s(C_m)$. The third entry in the table (the range of $s[C_{\mathcal{L}}(D, mQ_{\infty})]$) is $s(C_m) - \nabla^i(C_{\mathcal{L}}(D, mQ_{\infty}))$.

As well as those m for which $s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$, Table 4 also gives

$$(15) \quad s(C_m) = \nabla^i(C_{\mathcal{L}}(D, mQ_{\infty})) = s[C_{\mathcal{L}}(D, mQ_{\infty})]$$

for those $m \in J(n, g)$ such that

$$(16) \quad \begin{aligned} \frac{q-1}{2} - M^{\bullet} &\leq M^{\circ} \leq \frac{q-1}{2} && \text{if } q \text{ is odd,} \\ M^{\circ} &= q && \text{if } q \text{ is even.} \end{aligned}$$

Hence (15) also holds for those $m \in [\frac{n-1}{2} + g, \frac{n-3}{2} + 2g]$ such that m^{\perp} satisfies (16). In all these cases except $M^{\bullet} \geq 2$ and $M^{\circ} = \frac{q-3}{3}$ we have

$$s[C_{\mathcal{L}}(D, mQ_{\infty})] = s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_{\infty})) + 1.$$

For $M^{\bullet} \geq 2$ and $M^{\circ} = \frac{q-3}{3}$ we have

$$s[C_{\mathcal{L}}(D, mQ_{\infty})] = s(C_m) = \nabla(C_{\mathcal{L}}(D, mQ_{\infty})) + 2.$$

For q odd, this gives $\frac{q^2-1}{4}$ values of $m \in I(n, g)$ for which $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is determined but is strictly greater than $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$. Thus, for q odd, the total proportion of those m in $I(n, g)$ for which we have determined $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is

$$\frac{1}{2} + \frac{1}{2q} + \frac{q^2 - 1}{4(q^2 - q)} = \frac{3(q+1)}{4q}.$$

For q even, it gives $q-2$ values of $m \in I(n, g)$ for which $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is determined but is strictly greater than $\nabla(C_{\mathcal{L}}(D, mQ_{\infty}))$. Thus, for q even, the total proportion of those $m \in I(n, g)$ for which we have determined $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is

$$\frac{1}{2} + \frac{3q-5}{2(q^2-q-1)} + \frac{q-2}{q^2-q-1} = \frac{1}{2} + \frac{5q-9}{2(q^2-q-1)}.$$

Thus we have determined $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ for over three quarters of those m in $I(n, g)$ when q is odd but only for something over one half of those m in $I(n, g)$ when q is even. For q odd, the first m for which $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is not determined is $q = 5$ and $m = 70$ (when it is either 56 or 57), and for q even, the first m for which $s[C_{\mathcal{L}}(D, mQ_{\infty})]$ is not determined is $q = 8$ and $m = 268$ (when it is either 236 or 237).

Acknowledgment. We would like to thank Paddy Farrell for his continued interest in and support of our work.

REFERENCES

- [1] A.I. BARBERO AND C. MUNUERA, *The weight hierarchy of Hermitian codes*, SIAM J. Discrete Math., 13 (2000), pp. 79–104.
- [2] Y. BERGER AND Y. BE'ERY, *Trellis-oriented decomposition and trellis complexity of composite-length cyclic codes*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1185–1191.
- [3] T.D. BLACKMORE AND G.H. NORTON, *On the state complexity of some long codes*, in Finite Fields: Theory, Applications and Algorithms, R.C. Mullin and G.L. Mullen, eds., Contemp. Math. 225, AMS, Providence, RI, 1998, pp. 203–214.
- [4] T.D. BLACKMORE AND G.H. NORTON, *On the trellis structure of GRM codes*, in Proceedings of the Sixth International Workshop on Algebraic and Combinatorial Coding Theory, 1998, pp. 26–29.
- [5] T.D. BLACKMORE AND G.H. NORTON, *Lower bounds on the state complexity of geometric Goppa codes*, Des. Codes Cryptogr., to appear.
- [6] T.D. BLACKMORE AND G.H. NORTON, *On trellis structures for Reed-Muller codes*, Finite Fields Appl., 6 (2000), pp. 39–70.
- [7] T.D. BLACKMORE AND G.H. NORTON, *On a family of abelian codes and their state complexities*, IEEE Trans. Inform. Theory, 47 (2001), pp. 355–361.
- [8] T.D. BLACKMORE AND G.H. NORTON, *Bounds on the state complexity of geometric Goppa codes*, in Proceedings of the IEEE International Symposium on Information Theory, Sorrento, Italy, 2000, p. 170.
- [9] G.D. FORNEY, JR., *Dimension/length profiles and trellis complexity of linear block codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 1741–1752.
- [10] J. L. MASSEY, *Foundation and methods of channel encoding*, in Proceedings of the International Conference on Information Theory and Systems, 1978.
- [11] C. MUNUERA, *On the generalized Hamming weights of geometric Goppa codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 2092–2099.
- [12] R. PELLIKAAN, *On special divisors and the two variable zeta function of algebraic curves over finite fields*, in Arithmetic, Geometry and Coding Theory, Walter de Gruyter, Berlin, 1996, pp. 174–184; updated version available at <http://www.win.tue.nl/math/dw/personalpages/ruudp>.
- [13] Y. SHANY AND Y. BE'ERY, *Bounds on the state complexity of codes from the Hermitian function field and its subfields*, IEEE Trans. Inform. Theory, 46 (2000), pp. 1523–1527.
- [14] H. STICHTENOTH, *Algebraic Function Fields and Codes*, Springer-Verlag, Berlin, 1993.
- [15] M.A. TSFASMAN AND S.G. VLADUT, *Geometric approach to higher weights*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1564–1588.
- [16] K. YANG, P.V. KUMAR, AND H. STICHTENOTH, *On the weight hierarchy of geometric Goppa codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 913–920.

ON THE DISTRIBUTED COMPLEXITY OF COMPUTING MAXIMAL MATCHINGS*

MICHAŁ HAŃCOWIAK[†], MICHAŁ KAROŃSKI^{†‡}, AND ALESSANDRO PANCONESI[§]

Abstract. We show that maximal matchings can be computed deterministically in $O(\log^4 n)$ rounds in the synchronous, message-passing model of computation. This is one of the very few cases known of a nontrivial graph structure, and the only “classical” one, which can be computed distributively in polylogarithmic time without recourse to randomization.

Key words. graph algorithms, distributed, synchronous, deterministic

AMS subject classifications. 05C15, 05C85, 68R10

PII. S0895480100373121

1. Introduction. One of the fascinating questions of computer science is whether, and to what extent, randomization increases the power of algorithmic procedures. It is well known that, in general, randomization makes distributed algorithms more powerful, for there are examples of basic coordination tasks in asynchronous systems that cannot be solved by deterministic procedures but admit simple randomized solutions. Randomization is also demonstrably more powerful in synchronous systems, as shown by the important example of oblivious routing in the hypercube (see, for instance, [15, 21]). In this paper, we are interested in this question in the context of distributed graph algorithms, where a synchronous, message-passing network *without shared memory* is to compute a function of its own topology, and focus on the problem of computing maximal matchings. We show that maximal matchings can be computed in polylogarithmically many communication rounds by *deterministic* distributed algorithms. Therefore, as far as maximal matchings are concerned, randomization is not necessary to go over the sublinear “divide.”

To put our work into perspective, we review some of the relevant facts and literature.

In a distributed network or architecture without shared memory, the cost of sending a message between two nodes is proportional to their distance in the network. Since sending messages to faraway nodes is expensive, it is desirable that computation be based only on information available locally. This locality constraint can be quite severe when one is to compute a global function of input data that are spread across the network and represents a challenge from the point of view of algorithmic design. This communication problem is completely neglected in the popular PRAM model. There, the existence of a shared memory which can be accessed in unit time allows fast collection and dissemination of data among the processors. Once this assumption is removed and the cost of communication is taken into consideration,

*Received by the editors April 25, 2000; accepted for publication (in revised form) June 4, 2001; published electronically October 31, 2001.

<http://www.siam.org/journals/sidma/15-1/37312.html>

[†]Department of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland (karonski@amu.edu.pl, mhancow@amu.edu.pl). The research of these authors was partially supported by grant KBN 7T11C 032 20.

[‡]Department of Mathematics and Computer Science, Emory University, Atlanta, GA (michal@math.cs.emory.edu).

[§]DSI, La Sapienza, via Salaria 113, 00198 Roma, Italy (ale@dsi.uniroma1.it). The research of this author was supported by the Alexander von Humboldt Foundation.

several computational problems which were easily solvable suddenly become hard or unsolvable efficiently, especially if one is seeking *deterministic* solutions.

The study of distributed graph algorithms goes back (at least) to the work of Linial [17], where an $\Omega(\log^* n)$ lower bound for computing maximal independent sets (MISs) in the ring is given. Together with the $O(\log^* n)$ upper bound given by a beautiful algorithm of Cole and Vishkin [6], this is one of the all too rare examples in complexity theory where the complexity of a computational problem can be determined exactly (modulo constants). Interestingly, it can be shown that randomization does not help [22].

Generalizing from rings to bounded degree graphs, one sees that several classical graph structures of both theoretical and practical interest, including MISs, maximal matchings, $(\Delta + 1)$ - and even Δ -vertex colorings, can be computed in polylogarithmic time [1, 2, 7, 26]. (Δ denotes the maximum degree of the input network.) In fact, many of these algorithms are very satisfactory because they are both quite simple and really of low complexity, i.e., with small exponents and no hidden large constants.

Further generalizing from bounded degree graphs to general topologies has proven elusive, in spite of several efforts [1, 2, 18, 23, 26, 27]. The situation here is, more or less, as follows. For a reasonably large class of graph structures, the asymptotically best *deterministic* algorithm known to date uses $O(n^{\epsilon(n)})$ rounds, where $\epsilon(n)$ is a function which (very slowly) goes to 0 as n , the size of the network, grows. These solutions are mainly of theoretical interest, since the protocols are quite cumbersome, and their implementation would probably be prohibitively expensive. On the other hand, once randomization is allowed, the same graph structures can be computed in polylogarithmically many rounds. Furthermore, these randomized algorithms are usually extremely simple, and their actual complexity is very low. For instance, $(\Delta + 1)$ -vertex coloring and MIS can be computed in $O(\log n)$ rounds with high probability by exceedingly simple protocols [19, 20, 24].

Another important example of something that can be computed in polylogarithmic time by using randomness by distributed algorithms, while it is not known whether the same time bounds can be attained without it, is that of $(O(\log n), O(\log n))$ -decompositions, an interesting type of graph decomposition with several applications [1]. Using randomization, these structures can be computed in $O(\log^2 n)$ rounds [18]. In fact, there exist nontrivial functions, such as nearly optimal edge colorings, that can be computed, with high probability, by extremely simple, indeed trivial, randomized algorithms in $o(\log n)$ (little-oh of n) rounds or even, under suitable degree assumptions, in as few as $O(\log \log n)$ rounds [8].

The question then is whether, in the context of distributed graph algorithms, randomization is necessary in order to obtain protocols that run in polylogarithmically many rounds in the size of the network.

In an attempt to gain some insight into this problem, we show that for a nontrivial and important graph structure, maximal matchings, randomization is not needed. Matchings are important structures from a theoretical point of view but might also be of practical interest, since, in some situations, they correspond to a set of operations, say, data transfers, that can be performed simultaneously without mutual interferences. We note that maximal matching is a special case of the difficult open problem of determining whether MISs can be quickly computed deterministically in spite of the locality constraint. The solution presented here takes $O(\log^4 n)$ many rounds, improving an earlier result by the same authors [9]. Although the time complexity of our protocol is quite high, it should be remembered that even in the EREW-PRAM

model, where information can be distributed inexpensively via the shared memory, the best asymptotic complexity for computing maximal matchings is $O(\log^3 n)$ [11]. Our solution hinges on a distributed procedure for bipartite graphs which, for almost all vertices in the graph, cuts the degree of a vertex almost perfectly in half. This approximate degree splitter might be useful in other contexts.

To our knowledge, maximal matchings are one of the very few examples of nontrivial graph functions which can be computed deterministically in polylogarithmically many communication rounds in the distributed model without additional assumptions on the input network. Other notable exceptions are the so-called ruling forests of [1] and the k -dominating sets of [14], both of which, however, are not “classical” graph structures.

We end this section by spelling out our model of computation, the *synchronous, message-passing distributed network*. Here, a distributed network (or architecture) is modeled as an undirected graph. The vertices of the graph correspond to processors, and edges correspond to bidirectional communication links. The network is synchronous in the sense that computation takes place in a sequence of *rounds*; in each round, each processor reads messages sent to it by its neighbors in the graph, does any amount of local computation, and sends messages back to each of its neighbors. The time complexity of a distributed algorithm is then given by the number of rounds needed to compute the desired function. Each node of the network has a unique identifier (ID) and knows it. We assume that the IDs of the network are the integers from 1 to n .

The problem we study is this: A distributed network is to compute a maximal matching of its own (unknown) topology.

2. Preliminaries. We shall make use of standard graph theoretic terminology. In particular, given a set of edges X in a graph G , $G[X]$ denotes the subgraph of G induced by X with vertex set $V(G)$.

DEFINITION 2.1. *The weight of an edge uv in a graph G , denoted as $w_G(uv)$, is equal to the number of edges incident to it.*

DEFINITION 2.2. *Given a graph G and a set of edges $X \subseteq E(G)$, $\text{touch}(X)$ denotes the set of edges which are “touched” by X , i.e., the set X itself union the set of edges incident to some edge of X .*

A *degree-2 graph* is a graph G of maximum degree 2, i.e., a collection of paths and cycles. Let G be a degree-2 graph and suppose that it is equipped with a 3-vertex coloring. Then a maximal matching M of G can be computed in constant many rounds, as follows. Each vertex of color 1 selects one of its incident edges arbitrarily; let A be the set of edges thus selected. The set A induces a graph consisting of isolated edges and paths of length two. The middle point of each such path selects one of its two incident A -edges arbitrarily. This leaves a set $A' \subseteq A$ which is a matching with respect to G and a maximal matching with respect to (the graph induced by) A . The edges in $\text{touch}(A')$ are removed from G , leaving a leftover graph G' . Now, the same procedure is repeated with the vertices of color 2. They select a set B of edges, and from it a matching $B' \subseteq B$ is computed as before. Note that B' is disjoint from A' . Again, $\text{touch}(B')$ is removed from G' , leaving a graph G'' . This graph is itself a collection of isolated edges and paths of length two because all vertices that have not lost their two adjacent edges have color 3, and spaces between them are greater than 3. Therefore, a maximal matching C' (with respect to G'') can again be computed as before. Since $\text{touch}(C') \supseteq E(G'')$ there are no more edges left. The set $M := A' \cup B' \cup C'$ is therefore, by construction, a maximal matching with respect to

G .

In [7] and [6] two deterministic, distributed algorithms are given to compute a $(\Delta + 1)$ -vertex coloring of an input graph of maximum degree Δ , where Δ is constant. (The algorithm in [7] does not need this assumption.) Their time complexity is, respectively, $O(\Delta \log n)$ and $O(\Delta^\Delta \log^* n)$ many communication rounds, where n is the number of vertices of the input graph. Both algorithms are very simple in the sense that they can be easily implemented in our distributed model.

Altogether, this means that maximal matchings of degree-2 graphs can be computed in $O(\log n)$, or even $O(\log^* n)$, many communication rounds. (In fact, the result holds for any constant degree.) We record this fact for future reference.

FACT 2.3. *Maximal matchings of degree-2 graphs can be computed in $O(\log n)$ many communication rounds in the synchronous, message-passing model of computation.*

3. Reducing the problem to two-colored bipartite graphs. Suppose that we have a distributed procedure that, given a two-colored, bipartite graph, computes a matching that matches a constant fraction of the edges in $O(t(n))$ many rounds. Here and in the rest of the paper, two-colored means that the vertices know which side of the bipartition they belong to. In this section we show that, assuming this, there is a distributed procedure that, given any input graph, computes a maximal matching in $O(\log n t(n))$ many rounds.

We start with an outline of the algorithm which is meant to highlight the intuition behind our approach and, hopefully, to pave the way for the precise analysis to follow. The basic structure of the algorithm is standard. The maximal matching is computed incrementally in *phases*. The goal of a phase is to compute a matching \mathcal{M} such that $|\text{touch}(\mathcal{M})|$ is a constant fraction of $|E(G)|$. The edges of \mathcal{M} are added to the partial solution computed so far and are removed together with the edges incident to them. The leftover graph is the input to the next phase. This takes $O(\log n)$ phases since, by assumption, $|\text{touch}(\mathcal{M})|$ is a constant fraction of the number of edges of the input graph. The heart of the algorithm is a phase, which is as follows. A phase begins by computing an arbitrary orientation of the edges of the input graph G ; after that, every vertex splits itself into two vertices called *siblings*. One sibling inherits all of the incoming edges, while the other inherits all of the outgoing edges. We therefore obtain a bipartite graph \mathcal{B} whose sides of the bipartition are the “in” siblings and the “out” siblings. Notice that G and \mathcal{B} have the “same” edge set. In the bipartite graph \mathcal{B} , the orientation of the edges is ignored (or, if you prefer, removed) and, resorting to the abovementioned procedure, a matching \mathcal{L} is computed that matches a constant fraction of the edges of \mathcal{B} . Now, since the edge sets of the bipartite graph and the input graph coincide, in actuality \mathcal{L} matches a constant fraction of the edges of the entire input graph. The problem is that \mathcal{L} is a matching in \mathcal{B} but not in G ! The next step is to select a “real” matching $\mathcal{M} \subseteq \mathcal{L}$ such that $|\text{touch}(\mathcal{M})|$ is a constant fraction of $|\text{touch}(\mathcal{L})|$; i.e. \mathcal{M} still matches a constant fraction of the edges of G , but it is a true matching. This is achieved in the following way. A new, degree-2 graph C is generated from \mathcal{L} by “merging” each pair of siblings back together. C is a collection of paths and cycles, and its edge set is exactly \mathcal{L} . A matching \mathcal{M} is then computed inside C with the property that $|\text{touch}(\mathcal{M})|$ is a constant fraction of $|\text{touch}(E(C))|$, and therefore a constant fraction of $|E(G)|$, since $E(C)$ and \mathcal{L} are, essentially, the same set. To compute \mathcal{M} , a maximal matching J (as in junk) is computed in C . This can be done in $O(\log n)$ time by Fact 2.3. The edges in J are discarded, leaving a collection of isolated edges and paths of length two. From this, a matching \mathcal{M} is

computed by including into it all isolated edges and, for each path of length two, the edge that, in G , has the highest weight. This ends the phase.

A counting argument shows that \mathcal{M} matches a constant fraction of $|\text{touch}(\mathcal{L})|$, thereby matching a smaller, but still constant, fraction of $|E(G)|$. Therefore, $O(\log n)$ many phases suffice to compute a maximal matching in the entire graph.

After this outline, we present the algorithm more precisely and argue its correctness.

Procedure Match

$\mathcal{F} := \emptyset$ // \mathcal{F} will be the final matching

repeat

1. Direct every edge arbitrarily.
2. Each vertex u splits into two siblings, u_{in} and u_{out} . All incoming edges of u become (nonoriented) edges incident to u_{in} , while all outgoing edges of u become (nonoriented) edges incident to u_{out} . The resulting bipartite graph is denoted by \mathcal{B} .
3. Invoke procedure BIPARTITEMATCH, described in section 4, to compute a matching \mathcal{L} inside \mathcal{B} .
4. Compute a degree-2 graph C as follows: Each pair of siblings u_{in} and u_{out} is merged back into u . Edges of \mathcal{L} that were incident to u_{in} or u_{out} become edges incident to u .
5. Let D be the graph obtained from C by removing the vertices of degree one. Compute a maximal matching J of D . Consider the set of edges $E(C) - J$. This set consists of isolated edges and paths of length two. Each middle point of every such length-2 path removes the edge incident on itself that has the smallest weight in the current graph G_i . Let \mathcal{M} be the set of remaining edges.
6. Add \mathcal{M} to \mathcal{F} , the partial solution computed so far, remove \mathcal{M} and its neighboring edges, and repeat with the leftover graph as the input to the next phase, i.e.,

$$G_{i+1} := G_i[E(G_i) - \text{touch}(\mathcal{M})].$$

until the graph has no more edges.

In procedure MATCH, each vertex keeps executing the algorithm as long as its degree in the current graph G_i is at least one. Assuming that procedure BIPARTITEMATCH can be implemented in the distributed model, it is apparent that procedure MATCH is a distributed algorithm. We shall prove that the number of iterations of the repeat-until loop is $O(\log n)$. We remark that the vertices must know the value of n . The exact number of phases can be divined from Lemma 3.1, and from Lemmas 4.6 and 4.7, and is equal to $299 \log_2 n$ because $\text{touch}(\mathcal{M}) \geq \frac{1}{216}|E(G)|$ in each phase of the procedure MATCH where G is the input graph to that phase. In what follows, with slight abuse of notation we will identify $E(G)$ with $E(\mathcal{B})$ and \mathcal{L} with $E(C)$. Now let us argue the correctness of the algorithm. By construction, it is apparent that \mathcal{M} is a matching. The maximality of \mathcal{F} follows from the fact that, at the end of each phase, only the edges in $\text{touch}(\mathcal{M})$ are removed.

We now turn to the analysis of the running time. Since, by assumption, the matching \mathcal{L} returned by BIPARTITEMATCH “touches” a constant fraction of $|E(G)|$, the crux of the matter is to show that $|\text{touch}(\mathcal{M})|$ is a constant fraction of $|\text{touch}(\mathcal{L})|$. This would ensure that within $O(\log n)$ phases $\text{touch}(\mathcal{F}) = E(G)$; i.e., \mathcal{F} is maximal. We do this next.

LEMMA 3.1. *Let G be the graph input to the current phase of procedure MATCH, and let \mathcal{M} be the matching computed at the end of the phase. Assume that procedure BIPARTITEMATCH computes a matching \mathcal{L} of the bipartite graph \mathcal{B} such that*

$|touch(\mathcal{L})|$ is constant fraction of $|E(\mathcal{B})|$. Then \mathcal{M} matches a constant fraction of the edges of G .

Proof. Recall that $E(C) = \mathcal{L}$ and that J is a matching. Since, in step 5, the graph D is obtained from C by removing vertices of degree one,

$$touch(\mathcal{L}) = touch(E(C)) = touch(E(C) - J).$$

Then, since in step 5 each middle point selects the edge of largest weight,

$$|touch(\mathcal{M})| \geq \frac{|touch(E(C) - J)|}{2} = \frac{|touch(\mathcal{L})|}{2}.$$

The claim follows from the assumption that $|touch(\mathcal{L})|$ is a constant fraction of $|E(\mathcal{B})|$ and the fact that $E(G) = E(\mathcal{B})$. \square

The time complexity of a single phase of procedure MATCH can be estimated as follows. Step 3 takes $O(t(n)) = O(\log^3 n)$ many rounds, by hypothesis. Step 5 takes $O(\log n)$ many rounds, by Fact 2.3. All remaining steps take constant many rounds. Overall, we get

$$T(\text{MATCH}, n) = O(\log n \times t(n)) = O(\log^4 n)$$

many rounds for the whole of procedure MATCH. In the next section, we shall prove that the number of rounds for procedure BIPARTITEMATCH is indeed $O(\log^3 n)$, thereby proving the following theorem.

THEOREM 3.2. *On any given input graph G , procedure MATCH computes a maximal matching of G in $O(\log^4 n)$ many communication rounds in the synchronous, distributed model of computation. For the algorithm to work, each vertex must know the value of $n := |V(G)|$.*

4. Computing maximal matchings in two-colored bipartite graphs.

In this section, we show how to compute a maximal matching of a given bipartite graph $G(L, R, E)$, where, importantly, each vertex knows which side of the bipartition it belongs to. (L stands for left and R for right.) That is, adopting our terminology, G is two-colored. This assumption is verified by the bipartite graph \mathcal{B} generated by procedure MATCH. The basic idea of the algorithm is to partition the edges of the graph into subgraphs called blocks, to be defined more precisely later, and to compute “good” matchings inside each block, in parallel. The matchings computed in each block are then put together to form a new graph. Since the blocks are edge disjoint but not vertex disjoint, the new graph is not itself a matching. In order to select a “sizable” matching inside it, we adopt a greedy strategy. Roughly speaking, the blocks are ranked at the outset in such a way that an edge coming from a “high-order” block will match at least half as many edges as any subset of edges coming from any collection of “low-order” blocks. We now describe the solution in detail, starting with some definitions. Keep in mind that we are working inside a two-colored, bipartite graph $G = (L, R, E)$.

DEFINITION 4.1. $H_i = \{u \in L : \frac{n}{2^{i+1}} < d(u) \leq \frac{n}{2^i}\}$.

Remark. If we replaced n with $\Delta(G)$ in this definition, the resulting algorithm would have an improved running time (a $\log n$ factor replaced by a $\log \Delta$ factor), but this would require the additional knowledge of Δ . Notice that the H_i ’s, for $0 \leq i \leq \lfloor \log_2 n \rfloor$, partition the set L of left-hand-side vertices.

DEFINITION 4.2. *The block induced by H_i , denoted as B_i , is the subgraph of G induced by the edges incident to H_i . The vertex set of B_i is $H_i \cup N(H_i)$, where the latter is the set of neighbors of H_i . Obviously, $H_i \subseteq L$ and $N(H_i) \subseteq R$.*

The blocks B_i 's, for $0 \leq i \leq \lfloor \log_2 n \rfloor$, partition both $E(G)$ and the set of vertices L . Notice that a partition of G into blocks can be generated in constant time, provided that the value of n (or, alternatively, that of $\Delta(G)$) is known. Two different blocks, say, B_i and B_j , can have right-hand-side vertices in common. In the algorithm to follow, a matching is computed simultaneously in all blocks. If a vertex of R belongs to two or more blocks, it will virtually “duplicate” itself and participate in the computations of the matchings in all blocks to which it belongs.

To compute a matching in a block, edges are removed in phases until we are left with a matching. The matching has the property that if it is removed together with the edges incident to it, a constant fraction of the edges in the block will disappear. After these matchings are computed in parallel in all blocks, a final matching is computed that matches a constant fraction of the edges in the whole graph. Since a vertex $v \in R$ can participate in more blocks simultaneously, it is possible that at the end there is more than one edge incident to it, coming from matchings of different blocks. To select a final matching for the whole graph, an edge is selected greedily; the edge selected is the one whose insertion into the final matching would remove the maximum number of edges; it is the edge with the maximum number of incident edges or, in other words, the edge uv with the maximum value of $d(u)$. This simple locally greedy approach works; i.e., it results in a matching incident to a constant fraction of edges overall because of the way the sets H_i are defined. Namely, consider a sequence $i < j_1 < j_2 < \dots < j_k$; then the smallest degree of a vertex in H_i is at least half the sum of the degrees of vertices, each of which comes from H_{j_s} , $1 \leq s \leq k$. We now proceed to explain how a good matching inside a given block can be computed.

Given a subgraph $X \subseteq G(L, R, E)$, we shall use the shorthand notation $l(X) := V(X) \cap L$ and $r(X) := V(X) \cap R$. The degree of a vertex u inside X is denoted by $d_X(u)$. The next definition is pivotal.

DEFINITION 4.3. *An (a, d) -spanner of a block $B = (H_i, N(H_i), E_B)$ is a subgraph $S \subseteq B$ such that*

- $|l(S)| \geq a|H_i|$; i.e., S contains a constant fraction of the vertices of H_i ;
- for every $u \in l(S)$, $d_S(u) \in [1, d]$; i.e., the degree of left-hand-side vertices is constant; and,
- for every vertex $v \in r(S)$, $d_S(v) \leq 2^{-k(B)}d_B(v) + 1$, where $k(B) := \lfloor \log_2 n \rfloor - i - 4$.

A spanner is a subgraph of “small” degree that spans a “large” fraction of vertices in H_i . As we shall see, given a spanner in a block it is easy and inexpensive to compute a matching inside the spanner that “touches” a constant fraction of the edges of the block.

Roughly speaking, a spanner in a block is computed by repeatedly slashing the vertex degrees. In the process, we want to ensure that “almost all” of the vertices of H_i have their degree “almost perfectly” cut in half at each iteration of the algorithm. For vertices in H_i we ensure that the two conditions hold simultaneously: The number of edges lost is at least a half and, at the same time, for almost all of the vertices, it is not much less than that. For vertices in $N(H_i)$, on the other hand, we care only that the first requirement is satisfied. The third condition of the definition of a spanner states that all vertices, whether in H_i or $N(H_i)$, lose at least half of the edges at each iteration, so that after $k(B) := \lceil \log_2 D \rceil - 4$ many iterations, where $D := n/2^i$ is the upper bound on the maximum degree of vertices in H_i , the degree of every vertex in the spanner is at most 16.

In the next section, we shall see how to compute $(\frac{1}{2}, 16)$ -spanners in $O(\log^3 n)$

many rounds. In this section, we assume that this can be done by means of a procedure SPANNER. The algorithm for computing a matching in a block first computes a spanner in the block and then a matching by means of a simple two step procedure; each vertex of the left-hand side proposes one edge, and then every vertex of the right-hand side selects one among the proposed edges incident to itself.

DEFINITION 4.4. *The proposal graph of a spanner S , denoted by P , is any graph defined by the following procedure: Every vertex $v \in l(S)$ selects one of its incident edges arbitrarily. The graph P is the subgraph induced by the edges thus selected.*

Vertices in $r(S)$ which have no proposed edge incident to them do not belong to P . Therefore, $l(P) = l(S)$ and $r(P) \subseteq r(S)$. The latter inclusion can be strict. Note that P is a set of “stars” with the high degree on the right side.

DEFINITION 4.5. *The matching graph of a proposal graph P , denoted by M , is any graph defined by the following procedure: Every vertex $v \in r(P)$ selects one of its incident edges arbitrarily. The graph M is the subgraph induced by the edges thus selected.*

Clearly M is a matching. Also note that $r(M) = r(P)$ and $l(M) \subseteq l(P)$. The last inclusion can be strict. To summarize, here is the algorithm for selecting a matching M in a block $B = (H, N(H), E_B)$. The blocks are generated at the outset in constant time. For this to happen, every left-hand-side vertex needs to determine which set H_i it belongs to. This takes constant time, provided that every vertex knows the value of n (or, equivalently, the value of Δ). Recall that a right-hand-side vertex might participate to many blocks.

Procedure Matchblock

1. Let i be the input parameter. Compute a spanner S in B_i , the block defined by H_i , by invoking procedure SPANNER.
 2. Compute a proposal graph $P \subseteq S$ as follows: Every $v \in l(S)$ selects one incident edge arbitrarily. Vertices of $r(S)$ with no selected edges incident to them do not belong to P .
 3. Compute a matching $M \subseteq P$ as follows: Every vertex $u \in r(P)$ selects one incident edge arbitrarily.
-

The running time of procedure MATCHBLOCK is dominated by the number of rounds needed to compute a spanner; everything else takes constant time. Therefore, MATCHBLOCK takes

$$T(\text{MATCHBLOCK}, n) = T(\text{SPANNER}, n) = O(\log^3 n)$$

many rounds. The bound on the running time of SPANNER will be established in the next section.

After executing MATCHBLOCK in parallel in all blocks, we have a collection of matchings $\{M_i : 0 \leq i \leq \lfloor \log_2 n \rfloor\}$ with one matching in every block. Their union is not necessarily a matching, for vertices on the right-hand side might participate in more blocks and end up having degree higher than one. The final matching is determined by the following procedure.

Procedure GLOBALMATCH

Every vertex on the right-hand side selects, among the edges incident to itself, the edge of greatest weight $w_G(e)$.

To summarize, here is the overall algorithm for computing a matching that “touches” a constant fraction of the edges of a given two-colored, bipartite graph. (The proof of correctness will follow shortly.) Given two graphs X and Y , $X \oplus Y$ is the graph whose vertex and edge sets are, respectively, $V(X) \cup V(Y)$ and $E(X) \cup E(Y)$.

Procedure BipartiteMatch

1. Generate a partition of $G = (L, R, E)$ into blocks, where block B_i is induced by the set H_i .
 2. Compute a matching M_i inside every block B_i by invoking procedure MATCHBLOCK.
 3. Compute a matching \mathcal{M} inside the graph $\oplus_i M_i$ by invoking procedure GLOBALMATCH.
-

The overall running time of the procedure is dominated by the time needed by procedure MATCHBLOCK and is

$$T(\text{BIPARTITEMATCH}, n) = T(\text{MATCHBLOCK}, n) = O(\log^3 n).$$

The correctness of the procedure—that indeed \mathcal{M} is a matching in G —is straightforward. The next two lemmas show that $\text{touch}(\mathcal{M})$ contains a constant fraction of the edges of G .

LEMMA 4.6. *For every matching M computed by step 2 of procedure BIPARTITEMATCH, $\text{touch}(M)$ is a constant fraction of the edges of its block B .*

Proof. Let $m := |E(B)|$ and

$$D := \max_{u \in l(B)} \deg_B(u).$$

Fix any $b \in (0, 1)$ and let us consider two cases: Either

$$\sum_{v \in r(P)} d_B(v) \geq bm$$

or

$$\sum_{v \in r(P)} d_B(v) < bm.$$

For the first case, we know that $r(M) = r(P)$; therefore

$$|\text{touch}(M)| \geq bm.$$

For the second case, note that any proposal graph P satisfies the following conditions:

- $|l(P)| \geq a|l(B)|$.
- For every $u \in l(P)$, $d_P(u) = 1$.
- For every $v \in r(P)$, $d_P(v) \leq \frac{1}{2^{k(B)}} d_B(v) + 1$, where $k(B) := \lceil \log_2 D \rceil - 4$

and also

$$|l(B)| \frac{D}{2} \leq m \leq |l(B)| D,$$

since B is a block. Therefore, we can bound the set of left-hand-side matched vertices as follows:

$$\begin{aligned}
|l(M)| &= |l(P)| - \sum_{v \in r(P)} (d_P(v) - 1) \\
&\geq a|l(B)| - \sum_{v \in r(P)} (d_P(v) - 1) \\
&\geq a|l(B)| - \sum_{v \in r(P)} \frac{1}{2^{k(B)}} d_B(v) \\
&\geq a|l(B)| - \frac{1}{2^{k(B)}} bm \\
&\geq a|l(B)| - \frac{1}{2^{k(B)}} bD|l(B)| \\
&\geq (a - 16b)|l(B)|.
\end{aligned}$$

Recalling that for all $u \in l(B)$

$$\frac{D}{2} \leq d_B(u) \leq D,$$

we get the bound

$$\begin{aligned}
|\text{touch}(M)| &\geq |l(M)| \frac{D}{2} \\
&\geq \frac{a - 16b}{2} D|l(B)| \\
&\geq \frac{a - 16b}{2} m. \quad \square
\end{aligned}$$

LEMMA 4.7. *The set $\text{touch}(\mathcal{M})$, where \mathcal{M} is the matching computed by step 3 of procedure BIPARTITEMATCH, contains a constant fraction of edges of the input graph G .*

Proof. This follows from two observations. Recall that we are operating inside a bipartite graph $G = (L, R, E)$ whose left-hand side L is partitioned by the sets H_i , which are defined as the sets of vertices u such that $n/2^{i+1} < d_G(u) \leq n/2^i$ for $i \geq 0$. It follows that for every set of indices $i < j_1 < j_2 < \dots < j_m$, and no matter how we pick elements $u \in l(B_i) = H_i$ and $v_k \in H_{j_k}$,

$$d_G(u) \geq \frac{1}{2} \sum_{j \in J} d_G(v_j),$$

where $J := \{j_1, j_2, \dots, j_m\}$. Note that in the above summation there is a unique, but arbitrary, v_j for every H_j . The second observation is that procedure GLOBALMATCH selects the edge of biggest weight $w_G(\cdot)$. \square

Therefore, we obtain the following lemma.

LEMMA 4.8. *Procedure BIPARTITEMATCH, given in input a two-colored, bipartite graph G , produces a matching \mathcal{M} in $O(\log^3 n)$ many communication rounds such that $|\text{touch}(\mathcal{M})|$ is a constant fraction of the number of edges of G .*

5. How to compute a spanner. In this section, we shall show how to compute an (a, d) -spanner in a given block $B = (H, N(H), E)$. The basic intuition was outlined in section 4, but we repeat it here. The spanner is computed by approximately halving the degree of vertices of B . Recall that, by definition, in a block B_i the vertices of the left-hand side H_i have degree between $n/2^i$ and $n/2^{i+1}$. Consider the ideal situation in which the degree of every vertex is cut perfectly in half. Then there exists a k , with $k = O(\log n)$, such that after k iterations, we have a subgraph of B in which every left-hand vertex has degree between, say, 1 and 4. Israeli and Shiloach accomplish this much in the PRAM model of computation by resorting to Euler circuits [10]. Unfortunately, in our distributed model of computation the computation of Euler circuits requires a time proportional to the diameter of the network [25]. Therefore, a completely different approach is called for.

We start with some preliminaries. Given a set of edges F , consider a decomposition of $G[F]$ into a collection of cycles and paths, computed as follows. Each vertex of $G[F]$, in parallel, splits itself into vertices of degree two (pairing any two adjacent edges) and perhaps one vertex of degree one (if the degree of the original vertex is odd). The new vertices are called *siblings* and the original vertex is called the *parent*. A new graph is obtained with at least $|F|$ vertices and exactly $|F|$ edges. Such a decomposition is called here a *2-decomposition*. Note that a 2-decomposition can be generated in constant time in the distributed model of computation and that it consists solely of paths and cycles.

Now consider the problem of halving the degree of a parent. If we could remove exactly one edge incident to every sibling, parents whose degree is even would lose exactly half of their edges, while parents of odd degree would have their degree split almost perfectly in half. A possible approach to achieve this could be the following. Compute perfect or near perfect matchings in the paths and cycles of the 2-decomposition and remove the edges thus matched. In what follows, we will refer to both perfect and near perfect matchings with the acronym *nPMs*. Unfortunately, computing nPMs in a 2-decomposition requires linear time in the worst case [25]. To circumvent the problem, we “chop up” (partition) the 2-decomposition into smaller pieces and compute nPMs inside every piece, deleting the edges that are matched. This approach introduces errors because of “border” vertices, that is, vertices that belong to two adjacent pieces. The problem arises because a border vertex may be matched or unmatched from each “side” at the same time. If we decide to remove only matched edges, a parent that has many siblings as border vertices will not lose enough edges. Conversely, if we decide to remove all edges incident on border vertices, a parent might lose too many of them. This conundrum can be solved by a judicious choice of the length of the pieces with which the 2-decomposition is partitioned. These pieces should be neither too long for efficiency reasons, nor too short, so that the number of border vertices can be bounded. If we can do this, overall “most” parents will have their degree “almost perfectly” cut in half.

The above discussion motivates the following definitions. Given a partition of a 2-decomposition into connected components, the components are called *segments*. A segment is *long* if its length is at least

$$\ell := 100 \log^2 n$$

and *short* otherwise. The length of a segment is given by the number of edges it contains. A segment can be only an even cycle or a path. A vertex can possibly belong to two (adjacent) segments. Such vertices are called *border* vertices.

An edge adjacent to a border vertex is a *bad edge*. All other edges are *good*. Notice that edges adjacent to ends of paths are good.

DEFINITION 5.1. *A parent v is called pliable if it is adjacent to at most $d(v)/p$ bad edges. The value of p , the coefficient of pliability, is*

$$p := \log n.$$

All parents that are not pliable are called nasty.

Remark. We can now see why it is necessary to operate inside bipartite graphs. In an odd cycle, there is always going to be one sibling whose edges are unmatched. Conceivably, the 2-decomposition could consist entirely of triangles and, as a consequence, almost all parents would be nasty; i.e., after removal of matched and bad edges they would lose all of their incident edges. In a bipartite graph, on the other hand, cycles create no problems. When a cycle is short, a perfect matching can be computed inside it, while when it is partitioned into long segments it cannot create too many nasty parents.

We are ready to present our main procedure, called SPANNER, together with subroutines SPLITTER and LONGARROWS. SPANNER acts on a block, defined by the input parameter D . The block is the two-colored, bipartite graph $B = (H, N(H), E)$ defined by the set of left-hand vertices

$$H = \left\{ u : \frac{D}{2} < \deg(u) \leq D \right\}.$$

Clearly, given D , vertices can decide whether they belong to H locally. A bird's eye view of the algorithm is as follows. Starting with $B^0 := B$ and $P_0 := l(B^0) = H$, the algorithm generates a 2-decomposition of the block B_0 and partitions it into long segments. After computing nPMs inside the segments, both matched and bad edges are marked (removed).

Let $P_1, P_1 \subseteq P_0$, be the set of pliable vertices. These are the vertices whose degree has been split well. Next, both marked edges and nasty vertices (the set $P_0 \setminus P_1$) are removed from B^0 , yielding a new leftover block B^1 . Then a new 2-decomposition of B^1 is generated and partitioned into long segments. Again nPMs are computed inside the segments, and both matched and bad edges are marked. This defines a new set $P_2 \subseteq P_1$ of vertices that have been pliable twice in a row, and so on. Therefore, we obtain a sequence of subgraphs B^0, B^1, \dots, B^i and a sequence of subsets $P_0 \supseteq P_1 \supset \dots \supseteq P_i$ of pliable vertices, where $P_j = l(B^j)$.

Let us denote the degree of vertex v in B^j by $d_j(v)$. We will show that after k stages, where $k = O(\log D)$,

- (a) the size of P_k is a constant fraction of that of P_0 , and
- (b) for all $u \in P_k$, $d_k(u) = (1 \pm o(1))d_0(u)/2^k$.

Procedure SPANNER is as follows.

Procedure Spanner

1. Let $B^0 := B, P_0 := l(B^0)$.
 2. For $j := 0$ to $k = O(\log D)$ do:
 - (a) invoke procedure SPLITTER to mark edges of B^j ;
 - (b) each vertex of P_j , in parallel, enters the set P_{j+1} if it has less than $d_j(u)/p$ incident bad edges;
 - (c) remove marked edges and vertices $P_j \setminus P_{j+1}$ from B^j in order to get B^{j+1} .
-

The running time of SPANNER is

$$(5.1) \quad T(\text{SPANNER}, n) = O(\log D \times T(\text{SPLITTER}, n)).$$

Procedure SPLITTER acts on B^j , the current subgraph, by computing nPMs in the segments of the 2-decomposition in B^j and marking the resulting matched and bad edges. The segments, which are computed by procedure LONGARROWS (described below), will be directed: Each edge in the segment is given an orientation, and the orientation inside each segment is consistent; i.e., each segment is either a directed path or a directed cycle. The procedure to achieve this is described below. Here we notice that given an orientation in a two-colored graph, a nPM can be computed in constant time as follows: Let red and blue be the two colors; an edge enters the nPM if and only if its head points to a red vertex.

Procedure Splitter

1. Let B be the current subgraph. Each vertex of B , in parallel, generates its siblings, giving rise to a 2-decomposition of B .
 2. Procedure LONGARROWS(ℓ) is invoked to compute oriented segments in the 2-decomposition.
 3. In each segment, a nPM is computed using the given 2-vertex-coloring of B as follows: Let red and blue be the colors of the bipartition. Then each edge enters the matching if and only if its head points to a red vertex.
 4. Edges incident to border vertices, i.e., vertices separating two adjacent segments, enter the set of bad edges.
 5. All matched and bad edges are marked.
-

The running time of SPLITTER is

$$(5.2) \quad T(\text{SPLITTER}, n) = T(\text{LONGARROWS}, n).$$

Now let us turn to procedure LONGARROWS, the one responsible for partitioning the 2-decomposition into segments and computing nPMs inside them. We remark that such a partitioning could be computed by resorting to ruling sets [1], but the procedure given here is both simpler and more efficient.

Initially, each edge is given an arbitrary orientation. Each oriented edge is an *elementary arrow*. An *arrow* is a sequence of elementary arrows pointing to the same direction. Therefore, adjacent arrows always have contradictory directions. Given a path or an even cycle P of length k , procedure LONGARROWS(ℓ) partitions P into arrows of length at least ℓ . More specifically, if $\ell \geq k$, the whole of P becomes an arrow of length k ; otherwise, P is partitioned into arrows of length at least ℓ . This can be done optimally in time $O(\ell)$.

We distinguish two types of borders between arrows: “head to head” and “tail to tail.”

Procedure LongArrows

1. Let ℓ be the input parameter. At the beginning, all elementary arrows are set arbitrarily.
 2. For $i:=1$ to $\lceil \log \ell \rceil$ do

In parallel, for every border of type “head to head” do:

 - (a) if both arrows (touching the border) have length $< 2^i$, then choose one of them and reverse it (i.e., all of its elementary arrows are reversed).
 - (b) If an arrow has length $< 2^i$, then reverse it.
-

Note that we can check if an arrow has length $< 2^i$ in time $O(2^i)$ and we can reverse that arrow in time $O(2^i)$. It means that the whole algorithm takes

$$\sum_{i=1}^{\log \ell} O(2^i) = O(\ell)$$

rounds.

FACT 5.2. *Let C be a connected component of a 2-decomposition of a graph G . After $\text{LONGARROWS}(\ell)$ terminates, either C is an arrow or it is partitioned into arrows of length at least ℓ .*

Proof. Let k be the length of C and denote by $S(i)$ the sentence “at the end of the i th iteration all arrows have length $\geq 2^i$.” $S(0)$ is true. Let us assume $S(i)$ and prove $S(i+1)$ for $i < \log k$. At the beginning of iteration $i+1$, there are only “long arrows” of length $\geq 2^{i+1}$ and “short arrows” of length $\geq 2^i, < 2^{i+1}$. We have to prove that at the end of the current iteration all arrows are long. Every short arrow has to be in one of the following situations:

- (a) Its head touches the head of another short or long arrow; in both cases it will disappear.
- (b) Its head touches the end of the path; in this case, step 2(b) of procedure LONGARROWS will reverse it.

Therefore, after $\lceil \log \ell \rceil$ iterations either C is an arrow, if $k \leq 2\ell$, or it is partitioned into arrows of length at least ℓ . \square

We now proceed to analyze the behavior of procedure SPLITTER . In the next two facts, we shall describe the behavior of degrees of pliable vertices and the cardinality of the set of such vertices in a single call of the subroutine SPLITTER . In what follows, we shall keep the same notation as in the above procedures and use the following notation: By Δ_j and δ_j we denote the maximum and minimum degree in B^j of vertices from $P_j = l(B^j)$. The first fact says that the degree of pliable vertices is split almost perfectly. Essentially, it establishes conditions 2 and 3 of the definition of (a, d) -spanner. (The formal proof is given in Theorem 5.5.)

FACT 5.3. *For all $v \in P_{j+1}$,*

$$\frac{1}{2} \left(\left(1 - \frac{2}{\log n} \right) d_j(v) - 1 \right) \leq d_{j+1}(v) \leq \frac{1}{2} (d_j(v) + 1),$$

where j is any iteration of the for-loop of procedure SPANNER . The right-hand-side inequality holds for all $v \in V(B^j)$.

Proof. Let e_+ and e_- denote the number of good and bad edges, respectively, incident to a parent v . To bound $d_j(v)$ from above, note that the worst case occurs when v has no bad edges incident to itself. Therefore, when the $d_j(v)$ is odd and its

unique degree-1 sibling has no marked edge,

$$d_{j+1}(v) \leq \frac{1}{2}(e_+ + 1) = \frac{1}{2}(d_j(v) + 1).$$

This holds for all $v \in V(B^j)$.

For the lower bound, notice that a pliable parent v loses the largest number of edges if all of its siblings incident to bad edges have just one bad edge incident to them and have the other edge marked, so that both edges will be lost. Then, when v has odd degree,

$$d_{j+1}(v) \geq \frac{1}{2}(e_+ - e_- - 1) \geq \frac{1}{2} \left(\left(1 - \frac{2}{\log n}\right) d_j(v) - 1 \right),$$

since $e_- \leq d_j(v)/\log n$ by definition of pliable parent. \square

The next fact says that most vertices remain pliable from one iteration to the next, thereby establishing condition 1 of the definition of (a, d) -spanner. (Again, the formal proof of this claim is given in Theorem 5.5.)

FACT 5.4. *For all iterations $j = 0, \dots, k-1$ of procedure SPANNER,*

$$|P_{j+1}| \geq |P_j| \left(1 - \frac{2/100 \Delta_j}{\log n \delta_j}\right).$$

Proof. Let N_{j+1} be the set of nasty vertices at the end of iteration j , and let $be[N_{j+1}]$ be the number of bad edges incident to N_{j+1} . Notice that $|P_{j+1}| = |P_j| - |N_{j+1}|$.

A lower bound for $be[N_{j+1}]$ follows from the fact that vertices in N_{j+1} have, by definition, at least $\delta_j/\log n$ bad edges. Hence,

$$be[N_{j+1}] \geq |N_{j+1}| \frac{\delta_j}{\log n}.$$

Recall that our graph is bipartite and so all cycles (and in particular the short ones) of the 2-decomposition are of even length. Therefore, all bad edges arise from nPMs in long components (paths and cycles) only.

We can bound the number of bad edges incident to N_{j+1} by the total number of bad edges in B^j . Since $|E(B^j)| \leq \Delta_j |P_j|$, and since each nPM has length at least $\ell = 100 \log^2 n$ and contributes at most two bad edges,

$$be[N_{j+1}] \leq \frac{2|P_j|\Delta_j}{100 \log^2 n},$$

and the fact follows. \square

Finally, we shall present the analysis of procedure SPANNER.

THEOREM 5.5. *An invocation of procedure SPANNER with parameter D and with input graph $G = (V, E)$ computes an $(\frac{1}{2}, 16)$ -spanner with respect to G and the set $H := \{v \in V(G) : D/2 \leq \deg_G(v) \leq D\}$.*

Proof. We shall check whether the output graph of our procedure fulfills the three conditions it has to satisfy to be a spanner (Definition 4.3); that is, for $k := \lceil \log D \rceil - 4$,

- $|P_k| \geq \frac{1}{2}|P_0|$;
- for every $u \in P_k$, $d_k(u) \in [1, 16]$;
- for every $v \in r(B^k)$, $d_k(v) \leq 2^{-k}d_0(v) + 1$.

Fix $k \geq 1$ and note that $D/2 \leq d_0(v) \leq D$. By an easy induction, using Fact 5.3, we have that, for all $v \in P_k$,

$$q^k \left(\frac{D}{2} + 1 \right) - 1 \leq d_k(v) \leq \left(\frac{1}{2} \right)^k (D - 1) + 1,$$

where $q = (1 - 2/\log n)/2$. The upper bound, which also holds for $v \in r(B^k)$, establishes condition 3 of the definition of a spanner. For condition 2, if we set $k = \lceil \log D \rceil - c$, we obtain

$$q^k \left(\frac{D}{2} + 1 \right) - 1 > 2^{c-1} e^{-2(1+\frac{2}{\log n})} - 1$$

and

$$\left(\frac{1}{2} \right)^k (D - 1) + 1 < 2^c + 1.$$

Hence, choosing $c = 4$, vertices from the set P_k all have degrees belonging to the interval $[1, 16]$ for large enough n . This establishes the second condition.

Finally, we have to show that the first condition which determines the spanner also holds. That is,

$$|P_k| \geq \frac{1}{2} |P_0|.$$

Repeatedly applying the inequality established in Fact 5.4, we get

$$|P_k| \geq |P_0| \prod_{j=0}^{k-1} \left(1 - \frac{2/100 \Delta_j}{\log n \delta_j} \right).$$

However,

$$\frac{\Delta_j}{\delta_j} \leq \frac{2^{-j}(D-1)+1}{q^j(D/2+1)-1} \leq 16,$$

since the last fraction is an increasing function of j for $j \leq k$. Therefore, for n large enough,

$$|P_k| \geq |P_0| \left(1 - \frac{32/100}{\log n} \right)^{\log n} \geq e^{-33/100} |P_0| \geq \frac{1}{2} |P_0|. \quad \square$$

Acknowledgment. The third author gratefully acknowledges the hospitality of the Adam Mickiewicz University, where much of this work was done.

REFERENCES

- [1] B. AWERBUCH, A. V. GOLDBERG, M. LUBY, AND S. PLOTKIN, *Network decomposition and locality in distributed computing*, in Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, Research Triangle Park, NC, 1989, pp. 364–369.
- [2] B. AWERBUCH, B. BERGER, L. COWEN, AND D. PELEG, *Fast network decompositions*, in Proceedings of the ACM Symposium on Principles of Distributed Computing, 1992, pp. 169–177.

- [3] N. ALON, J. SPENCER, AND P. ERDŐS, *The Probabilistic Method*, John Wiley and Sons, New York, 1992.
- [4] B. BOLLOBÁS, *Graph Theory*, Springer-Verlag, New York, 1979.
- [5] B. BOLLOBÁS, *Chromatic number, girth, and maximal degree*, *Discrete Math.*, 24 (1978), pp. 311–314.
- [6] R. COLE AND U. VISHKIN, *Deterministic coin tossing with applications to optimal parallel list ranking*, *Inform. and Control*, 70 (1986), pp. 32–53.
- [7] A. V. GOLDBERG, S. A. PLOTKIN, AND G. E. SHANNON, *Parallel symmetry-breaking in sparse graphs*, *SIAM J. Discrete Math.*, 1 (1988), pp. 434–446.
- [8] D. A. GRABLE AND A. PANCONESI, *Nearly optimal distributed edge colouring in $O(\log \log n)$ rounds*, *Random Structures Algorithms*, 10 (1997), pp. 385–405.
- [9] M. HAŃCOWIAK, M. KAROŃSKI, AND A. PANCONESI, *On the distributed complexity of computing maximal matchings*, in *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 1998, pp. 219–225.
- [10] A. ISRAELI AND Y. SHILOACH, *An improved algorithm for maximal matching*, *Inform. Process. Lett.*, 22 (1986), pp. 57–60.
- [11] Y. HAN, *An improvement on parallel computation of a maximal matching*, *Inform. Process. Lett.*, 56 (1995), pp. 343–348.
- [12] H. J. KARLOFF AND D. B. SHMOYS, *Efficient parallel algorithms for edge coloring problems*, *J. Algorithms*, 8 (1987), pp. 39–52.
- [13] R. M. KARP, *Probabilistic recurrence relations*, in *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, New Orleans, LA, 1991, pp. 190–197.
- [14] S. KUTTEN AND D. PELEG, *Fast distributed construction of k -dominating sets and applications*, in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, Ottawa, Ontario, 1995, pp. 238–249.
- [15] N. LYNCH, *Distributed Algorithms*, Morgan Kaufmann, San Francisco, 1996.
- [16] A. PANCONESI, *Lecture Notes in Distributed Algorithms*, <http://www.nada.kth.se/kurser/kth/2D5340>.
- [17] N. LINIAL, *Locality in distributed graph algorithms*, *SIAM J. Comput.*, 21 (1992), pp. 193–201.
- [18] N. LINIAL AND M. SAKS, *Low diameter graph decomposition*, *Combinatorica*, (1993), pp. 441–454.
- [19] M. LUBY, *A simple parallel algorithm for the maximal independent set problem*, in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, Providence, RI, 1985, pp. 1–10.
- [20] M. LUBY, *Removing randomness in parallel without processor penalty*, *J. Comput. System Sci.*, 47 (1993), pp. 250–286.
- [21] R. MOTWANI AND P. RAGHAVAN, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.
- [22] M. NAOR, *A lower bound on probabilistic algorithms for distributive ring coloring*, *SIAM J. Discrete Math.*, 4 (1991), pp. 409–412.
- [23] M. NAOR AND L. STOCKMEYER, *What can be computed locally?*, *SIAM J. Comput.*, 24 (1995), pp. 1259–1277.
- [24] Ö. JOHANSSON, *private communication*.
- [25] A. PANCONESI, *Lecture Notes in (Theoretical) Distributed Computing*, KTH NADA Tech. Report TRITA-NA-9801, KTH, Stockholm, Sweden, 1998.
- [26] A. PANCONESI AND A. SRINIVASAN, *The local nature of Δ -coloring and its algorithmic applications*, *Combinatorica*, 15 (1995), pp. 255–280.
- [27] A. PANCONESI AND A. SRINIVASAN, *On the complexity of distributed network decomposition*, *J. Algorithms*, 20 (1996), pp. 356–374.

CONSTRUCTING WORST CASE INSTANCES FOR SEMIDEFINITE PROGRAMMING BASED APPROXIMATION ALGORITHMS*

NOGA ALON[†], BENNY SUDAKOV[‡], AND URI ZWICK[§]

Abstract. Semidefinite programming based approximation algorithms, such as the Goemans and Williamson approximation algorithm for the MAX CUT problem, are usually shown to have certain performance guarantees using local ratio techniques. Are the bounds obtained in this way tight? This problem was considered before by Karloff [*SIAM J. Comput.*, 29 (1999), pp. 336–350] and by Alon and Sudakov [*Combin. Probab. Comput.*, 9 (2000), pp. 1–12]. Here we further extend their results and show, for the first time, that the local analyses of the Goemans and Williamson MAX CUT algorithm, as well as its extension by Zwick, are tight for every possible relative size of the maximum cut in the sense that the expected value of the solutions obtained by the algorithms may be as small as the analyses ensure. We also obtain similar results for a related problem. Our approach is quite general and could possibly be applied to some additional problems and algorithms.

Key words. MAX CUT, semidefinite programming, approximation algorithm

AMS subject classifications. 68W25, 90C22, 90C27, 05C85

PII. S0895480100379713

1. Introduction. MAX CUT is one of the most natural combinatorial optimization problems. An instance of MAX CUT is a graph. The goal is to partition the vertices of the graph into two sets such that the number, or the total weight, of the edges that cross the cut formed by this partition is maximized. Goemans and Williamson [GW95] describe an elegant approximation algorithm for the MAX CUT problem and show that its performance guarantee is at least $\alpha = \min_{0 < \theta < \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta} = 0.87856\dots$. No polynomial time approximation algorithm for MAX CUT can have a performance ratio of more than $\frac{16}{17}$, unless P=NP (Håstad [Hås97], Trevisan et al. [TSSW96]).

The MAX CUT approximation algorithm of Goemans and Williamson [GW95] uses a *semidefinite programming* relaxation of the problem. In this relaxation, every vertex i of the graph has a unit vector $v_i \in R^n$ associated with it. The algorithm solves this relaxation and then uses a simple randomized rounding technique to convert the constellation of unit vectors obtained into a cut. To get a lower bound on the performance ratio of the algorithm, Goemans and Williamson consider the worst possible ratio between the probability that a given edge is in the cut and the contribution of that edge to the optimal value of the semidefinite program. This worst case local ratio is attained when the angle θ between the two vectors v_i and v_j that correspond to the two endpoints of the edge is equal to $\theta_0 = \operatorname{argmin}_{0 < \theta < \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta} \simeq 2.331122\dots$.

*Received by the editors October 18, 2000; accepted for publication (in revised form) November 22, 2001; published electronically January 4, 2002. A preliminary version of this paper appeared in *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001, ACM, New York, 2001, pp. 92–100.

<http://www.siam.org/journals/sidma/15-1/37971.html>

[†]Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel (noga@math.tau.ac.il). The research of this author was supported in part by a USA Israeli BSF grant and by a grant from the Israel Science Foundation.

[‡]Department of Mathematics, Princeton University, Princeton, NJ 08540 and Institute for Advanced Study, Princeton, NJ 08540 (bsudakov@math.princeton.edu). The research of this author was supported in part by NSF grants DMS-0106589 and CCR-9987845 and by the state of New Jersey.

[§]Department of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel (zwick@cs.tau.ac.il).

Is the local analysis of the MAX CUT approximation algorithm of Goemans and Williamson [GW95] globally tight? In other words, are there graphs for which the optimal value of the relaxation is equal to the size of the maximum cut and for which there is an optimal solution of the relaxation in which the angle between every two vectors that correspond to vertices in the graph that are connected by an edge is exactly, or very close to, θ_0 ? Karloff [Kar99] was the first to construct graphs that satisfy these conditions and therefore show that the local analysis of the MAX CUT approximation algorithm of Goemans and Williamson is indeed tight. Karloff's result was simplified by Alon and Sudakov [AS00].

Goemans and Williamson [GW95] give a better lower bound on the performance guarantee of their algorithm for graphs that have relatively large cuts. More specifically, for a graph $G = (V, E)$, let A be the *relative size* of the maximum cut of G , i.e., the ratio between the size (or weight) of the maximum cut and the number of edges (or total weight of the edges) of G . Note that $1/2 < A \leq 1$. It is shown in [GW95] that if $A > t_0 \simeq 0.84458$, where $t_0 = \operatorname{argmin}_{0 < t \leq 1} h(t)/t$ and $h(t) = \arccos(1 - 2t)/\pi$, then the performance ratio of the MAX CUT algorithm is at least $\alpha(A) = h(A)/A > \alpha$. Karloff [Kar99] and Alon and Sudakov [AS00] show that this lower bound is again tight for every $t_0 \leq A \leq 1$.

What happens on graphs with $1/2 < A < t_0$? Goemans and Williamson [GW95] can show only that the performance ratio of their algorithm on such graphs is at least α . Zwick [Zwi99] presents a modification of the algorithm of Goemans and Williamson [GW95] that has a performance guarantee $\alpha'(A)$ strictly larger than $\alpha \simeq 0.87856$ for any $1/2 < A < t_0$. Furthermore, $\alpha'(A)$ approaches 1 as A decreases towards $1/2$.

In this paper we show, among other things, that the local analysis of the algorithms of Goemans and Williamson [GW95] and of Zwick [Zwi99] in the range $1/2 < A \leq t_0$ is again tight. Showing that the analysis of the MAX CUT algorithm is tight in the range $1/2 < A \leq t_0$ is a more challenging task than the corresponding task for the range $t_0 \leq A \leq 1$. To accomplish this task we construct, for any rational $-1 < \eta < 0$ and any rational $\frac{1}{2} < a \leq \frac{1-\eta}{2}$, a graph $G = (V, E)$ for which the size of the maximum cut is exactly $a|E|$, for which the optimal value of the relaxation is also equal to $a|E|$, and for which there is an optimal solution v_1, v_2, \dots, v_n of the relaxation such that for every $\{i, j\} \in E$ we have either $v_i \cdot v_j = \eta$ or $v_i \cdot v_j = 1$. (Note that the requirement that the value of the relaxation be $a|E|$ determines the proportion of the edges for which the inner product should be $v_i \cdot v_j = \eta$.)

The graphs used by Alon and Sudakov [AS00] to show that the analysis of the MAX CUT algorithm is tight in the range $t_0 \leq A \leq 1$ are graphs arising from *Hamming association schemes* over the binary alphabet. (Karloff [Kar99] uses the related Kneser graphs.) The graphs we use here to show that the analysis is also tight in the range $\frac{1}{2} < A \leq t_0$ are obtained by composing Hamming graphs and *expander* graphs. More specifically, if H is an appropriate Hamming graph and B is an appropriate bipartite expander with b vertices on each of its sides, then the graph that we use is obtained by replacing each vertex of H by a clique on b vertices and replacing each edge of H by a copy of B .

We believe that the technique developed in this paper could be used to construct worst case instances for other semidefinite programming based approximation algorithms. To demonstrate it, we use our technique to show that local analysis of the MAX NAE-3-SAT algorithm of Zwick [Zwi99] is also tight. This is an even more demanding task, as will be explained later.

An instance of MAX NAE- $\{3\}$ -SAT in the variables x_1, x_2, \dots, x_n is a weighted collection of triplets of the form $\langle z_1, z_2, z_3 \rangle$, where each z_i is a *literal*, i.e., one of the variables x_1, x_2, \dots, x_n or a negation of one of the variables, and the weights are nonnegative. The three literals appearing in a triplet must be distinct. A triplet (z_1, z_2, z_3) is satisfied by an assignment of 0-1 values to the variables x_1, x_2, \dots, x_n if at least one of the literals in the triplet is assigned the value 0 and at least one is assigned the value 1. MAX NAE- $\{3\}$ -SAT is an interesting problem as it can be seen as a generalization of both MAX CUT and of the problem of finding a maximum cut in 3-uniform hypergraphs.

The rest of this paper is organized as follows. In the next section we quickly review the MAX CUT approximation algorithm of Goemans and Williamson [GW95] and its extension by Zwick [Zwi99]. In section 3 we present the construction of the graphs that show that the local analysis of the MAX CUT algorithms of [GW95] and [Zwi99] are tight. In section 4 we review the MAX NAE- $\{3\}$ -SAT approximation algorithm of Zwick [Zwi99]. In section 5 we modify the construction of section 3 to show that the local analysis of the MAX NAE- $\{3\}$ -SAT approximation algorithm is again tight. We end in section 6 with some concluding remarks and open problems.

It is worth noting that our results here merely show that the analyses of the algorithms discussed are tight and do not exclude the possibility that these algorithms (or some variants of them) may have a better performance either by showing that with nonnegligible probability the rounding will output a solution that exceeds the expectation significantly or by proving that one can obtain other solutions to the corresponding semidefinite programs, solutions that may behave better in the rounding phase. Yet, the results here do show that some essentially novel ideas will be needed in order to improve the performance guarantees of the algorithms discussed.

2. Approximation algorithms for the MAX CUT problem. Let $G = (V, E)$ be a graph, where $V = \{1, \dots, n\}$. We let $OPT(G)$ denote the size of the maximum cut of G . The Goemans and Williamson approximation algorithm for MAX CUT starts by solving the following semidefinite programming relaxation of the problem:

$$\max_{\|v_i\|^2=1} \sum_{\{i,j\} \in E} \frac{1 - v_i^t v_j}{2},$$

where each v_i ranges over all n -dimensional unit vectors. (All our vectors are considered to be column vectors, and hence $v^t u$ is simply the inner product of v and u .) It is easy to see that the optimal value z^* of this program is at least as large as $OPT(G)$, the size of the maximum cut of G .

The algorithm of Goemans and Williamson [GW95] then rounds an optimal solution v_1, \dots, v_n of the semidefinite program by choosing a random unit vector r and defining $S = \{i \mid r^t v_i \leq 0\}$. This supplies a cut $(S, V-S)$ of the graph G . Let W denote the size of the random cut produced in this way and let $E[W]$ be its expectation. By linearity of expectation, the expected size is the sum, over all $\{i, j\} \in E$, of the probabilities that the vertices i and j lie in opposite sides of the cut. This last probability is precisely $\arccos(v_i^t v_j) / \pi$. Thus the expected value of the weight of the random cut is exactly $\sum_{\{i,j\} \in E} \frac{\arccos(v_i^t v_j)}{\pi}$. However, the optimal value z^* of the semidefinite program is equal to $z^* = \sum_{\{i,j\} \in E} \frac{1 - v_i^t v_j}{2}$. Therefore the ratio between

$E[W]$ and the optimal value z^* satisfies

$$\frac{E[W]}{z^*} = \frac{\sum_{\{i,j\} \in E} \arccos(v_i^t v_j^t)/\pi}{\sum_{\{i,j\} \in E} (1 - v_i^t v_j^t)/2} \geq \min_{\{i,j\} \in E} \frac{\arccos(v_i^t v_j^t)/\pi}{(1 - v_i^t v_j^t)/2}.$$

Denote $\alpha = \min_{0 < \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos \theta}$. An easy computation shows that the minimum α is attained at $\theta = \theta_0 = 2.3311\dots$, the nonzero root of $\cos \theta + \theta \sin \theta = 1$, and that $\alpha \in (0.87856, 0.87857)$. Thus, $E[W] \geq \alpha \cdot z^*$, and since the value of z^* is at least as large as the weight OPT of the maximum cut, we conclude that $E[W] \geq \alpha \cdot OPT$. It follows that the Goemans–Williamson algorithm supplies an α -approximation for MAX CUT. Moreover, by the above discussion, the expected size of the cut produced by the algorithm is not better than $\alpha \cdot OPT$ if $OPT = z^*$ and if v_1, \dots, v_n is an optimal solution of the semidefinite program that satisfies $\frac{\arccos(v_i^t v_j^t)/\pi}{(1 - v_i^t v_j^t)/2} = \alpha$ for every $\{i, j\} \in E$.

If the value of the semidefinite program is a large fraction of the total number of edges of G , the above reasoning, together with a simple convexity argument, is used in [GW95] to show that the performance of the algorithm is better. Let $h(t) = \arccos(1 - 2t)/\pi$ and let t_0 be the value of t for which $h(t)/t$ attains its minimum in the interval $(0, 1]$. Then t_0 is approximately 0.84458. Define $a = z^*/|E|$. If $a \geq t_0$, then, as shown in [GW95], $E[W] \geq \frac{h(a)}{a} z^* \geq \frac{h(a)}{a} OPT$. Note that $A = OPT/|E| \leq a$. As $h(a)/a$ is an increasing function of a , for $A \geq t_0$, we have also that $E[W] \geq \frac{h(A)}{A} OPT$. Here, as before, the actual expected size of the cut produced by the algorithm is not better than $\frac{h(a)}{a} OPT$ if $OPT = z^*$ and if v_1, \dots, v_n is an optimal solution of the semidefinite programming problem that satisfies $v_i^t v_j^t = 1 - 2a$ for every $\{i, j\} \in E$.

Karloff [Kar99] and Alon and Sudakov [AS00] showed that the analysis of the algorithm of Goemans and Williamson [GW95] is tight for every $t_0 \leq a \leq 1$. More precisely, for any rational $t_0 \leq a \leq 1$, there are infinitely many graphs for which the size of the maximum cut OPT is equal to z^* and also $E[W] = (h(a)/a)z^* = (h(a)/a)OPT$. In the next section we extend this result even further and show that the analysis of the algorithm of Goemans and Williamson is tight for all $1/2 \leq a \leq 1$.

To show that the analysis of Goemans and Williamson [GW95] is also tight in the range $1/2 \leq a \leq t_0$, we construct, for every rational a in this range, an infinite sequence of graphs for which the size OPT of the maximum cut and the optimal value z^* of the relaxation are both $a|E|$ and for which the relaxation has an optimal solution v_1, v_2, \dots, v_n such that for every $\{i, j\} \in E$ we have either $v_i^t v_j^t = \cos \theta_0$ or $v_i^t v_j^t = 1$. Indeed, the randomized rounding for such a solution satisfies $E[W] = \frac{h(t_0)}{t_0} z^* = \frac{h(t_0)}{t_0} OPT$, as for any edge ij for which $\frac{1 - v_i^t v_j^t}{2} \neq 0$ we have $v_i^t v_j^t = \cos \theta_0$.

Zwick [Zwi99] describes a modification of the algorithm of Goemans and Williamson [GW95] that has a better performance guarantee in the range $1/2 \leq a \leq t_0$. His algorithm works as follows. After solving the relaxation and obtaining a , which is assumed to satisfy $a < t_0$, the algorithm finds the unique solutions $c = c(a)$ and $t = t(a)$ of the following two equations:

$$\frac{\arccos(c(1 - 2t)) - \arccos(c)}{t} = \frac{2c}{\sqrt{1 - c^2(1 - 2t)^2}}, \quad \frac{1 - \frac{t}{a}}{\sqrt{1 - c^2}} = \frac{1 - 2t}{\sqrt{1 - c^2(1 - 2t)^2}}.$$

The algorithm then constructs a sequence of unit vectors w_1, w_2, \dots, w_n such that $w_i^t w_j^t = c(v_i^t v_j^t)$ for every $i \neq j$. The vectors w_1, w_2, \dots, w_n , and not the vectors

v_1, v_2, \dots, v_n , are then rounded using a random hyperplane. It is shown in [Zwi99], using local analysis, that the performance ratio achieved by this algorithm is at least

$$\alpha'(a) = \left(\frac{1}{a} - \frac{1}{t(a)} \right) h_{c(a)}(0) + \frac{h_{c(a)}(t(a))}{t(a)},$$

where $h_c(t) = \arccos(c(1-2t))/\pi$. It is also shown there that this analysis is tight if the size of the maximum cut in the graph is $a|E|$ and if for every $\{i, j\} \in E$ we have either $v_i^t v_j = 1 - 2t(a)$ or $v_i^t v_j = 1$. It is not difficult to see that $a < t(a)$ for every $1/2 < a < t_0$.

3. Worst case instances for the MAX CUT algorithms. In this section we prove the following theorem.

THEOREM 3.1. *Let $-1 < \eta < 0$ and $\frac{1}{2} < a \leq \frac{1-\eta}{2}$ be rational numbers. Then, for infinitely many values of n there exists a graph $G = (V, E)$, $V = \{1, \dots, n\}$ and a sequence u_1, u_2, \dots, u_n of unit vectors such that either $u_i^t u_j = \eta$ or $u_i^t u_j = 1$ for all $\{i, j\} \in E$, and the size of maximum cut in G is equal to*

$$\max_{\|v_i\|^2=1, v_i \in R^n} \sum_{\{i,j\} \in E} \frac{1 - v_i^t v_j}{2} = \sum_{\{i,j\} \in E} \frac{1 - u_i^t u_j}{2} = a|E|.$$

By the discussion in the previous section, it follows that the analyses of the algorithms of Goemans and Williamson [GW95] and of Zwick [Zwi99] are tight also in the range $1/2 \leq a \leq t_0$.

To prove Theorem 3.1 we first need to establish a connection between the smallest eigenvalue of a graph and the semidefinite relaxation of the MAX CUT problem. This is done in the following well-known lemma, whose proof we include here for the sake of completeness.

LEMMA 3.2. *Let G be a multigraph on the set $V = \{1, 2, \dots, n\}$, with adjacency matrix $A = (a_{ij})$, where a_{ij} corresponds to the multiplicity of the edge between i and j . Let $\lambda_1 \geq \dots \geq \lambda_n$ be the eigenvalues of $A = (a_{ij})$. Then*

$$\sum_{i < j} a_{ij} \frac{1 - v_i^t v_j}{2} \leq \frac{1}{2}|E(G)| - \frac{1}{4}\lambda_n \cdot |V(G)| = \frac{1}{2}|E(G)| - \frac{1}{4}\lambda_n \cdot n$$

for any set v_1, \dots, v_n of unit vectors in R^k , $k > 0$. In addition let $B = (b_{ij})$ be the $n \times k$ matrix whose rows are the vectors v_1^t, \dots, v_n^t . Then equality holds if and only if each column of B is an eigenvector of A with eigenvalue λ_n .

Note that for every loopless graph G with edges, $\lambda_n < 0$, as the sum of the eigenvalues is the trace of A , which is 0.

Proof. Let y_1, \dots, y_k be the columns of B . By definition we have $\sum_{i=1}^k \|y_i\|^2 = \sum_{ij} b_{ij}^2 = \sum_{i=1}^n \|v_i\|^2 = n$. Therefore

$$\sum_{i < j} a_{ij} \frac{1 - v_i^t v_j}{2} = \frac{1}{2}|E| - \frac{1}{2} \sum_{i < j} a_{ij} v_i^t v_j = \frac{1}{2}|E| - \frac{1}{4} \sum_{i=1}^k y_i^t A y_i.$$

By the variational definition of the eigenvalues of A , for any vector $z \in R^n$, $z^t A z \geq \lambda_n \|z\|^2$ and equality holds if and only if $Az = \lambda_n z$. This implies that

$$\sum_{i < j} a_{ij} \frac{1 - v_i^t v_j}{2} \leq \frac{1}{2}|E| - \frac{1}{4}\lambda_n \sum_{i=1}^k \|y_i\|^2 = \frac{1}{2}|E| - \frac{1}{4}\lambda_n \cdot n.$$

Equality holds in the last expression if and only if each y_i is an eigenvector of A with eigenvalue λ_n . \square

The main ingredient of our constructions are graphs arising from the Hamming association scheme over the binary alphabet. Let $V = \{v_1, v_2, \dots\}$ be the set of all vectors of length m over the alphabet $\{-1, +1\}$. For any two vectors $x, y \in V$ denote by $d(x, y)$ their *Hamming distance*, that is, the number of coordinates in which they differ. The *Hamming graph* $H = H(m, b)$ is the graph whose vertex set is V and in which two vertices $x, y \in V$ are adjacent if and only if $d(x, y) = b$. Here we consider only even values of b which are greater than $m/2$. We may and will assume, whenever this is needed, that m is sufficiently large.

Consider any two adjacent vertices of $H(m, b)$, v_i , and v_j . By the definition of H , the inner product $v_i^t v_j$ is $m - 2b$. Choose m and b such that $b > m/2$ is even and $\frac{m-2b}{m} = \eta$. This is always possible since η is a rational number, $-1 < \eta < 0$. Let $w_i = \frac{1}{\sqrt{m}}v_i$ for all i ; thus $\|w_i\|^2 = 1$ and $w_i^t w_j = \eta$ for any pair of adjacent vertices.

Note that by definition, $H(m, b)$ is the Cayley graph of the multiplicative group $Z_2^m = \{-1, +1\}^m$ with respect to the set U of generators formed by all vectors with exactly b coordinates equal to -1 , where vectors in the group multiply coordinate-wise. Therefore (see, e.g., [Lov93], Problem 11.8 and the hint to its solution) the eigenvectors of the adjacency matrix of $H(m, b)$ are the multiplicative characters χ_I of Z_2^m , where $\chi_I(x) = \prod_{i \in I} x_i$, and I ranges over all subsets of $\{1, \dots, m\}$. The eigenvalue corresponding to χ_I is $\sum_{x \in U} \chi_I(x)$. The eigenvalues of H are thus equal to the so-called *binary Krawtchouk polynomials* (see [CHLL97])

$$(3.1) \quad P_b^m(k) = \sum_{j=0}^k (-1)^j \binom{k}{j} \binom{m-k}{b-j}, \quad 0 \leq k \leq m.$$

The eigenvalue $P_b^m(k)$ corresponds to the characters χ_I with $|I| = k$ and thus has multiplicity $\binom{m}{k}$. Since $H(m, b)$ is a regular graph with degree $d = \binom{m}{b}$, its largest eigenvalue is equal to d and its corresponding eigenvector is $(1, 1, \dots, 1)$. In addition it was proved in [AS00] that if m is big enough, then the smallest eigenvalue of $H(m, b)$ is $\lambda = P_b^m(1) = \frac{m-2b}{m} \binom{m}{b}$. By the above discussion this eigenvalue has multiplicity $\binom{m}{1} = m$ and eigenvectors y_1, \dots, y_m with ± 1 coordinates, where for each vertex $v_j = (v_{j1}, \dots, v_{jm})$, $y_i(v_j) = v_{ji}$. Therefore the columns of the matrix, whose rows are the vectors w_i , are the eigenvectors $\frac{1}{\sqrt{m}}y_i$ of $A(H)$ corresponding to the eigenvalue λ .

Let $A = (a_{ij})$ be an $s \times s$ matrix and $B = (b_{pq})$ be a $t \times t$ matrix; then the *tensor product* of A and B is the $st \times st$ matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1s}B \\ a_{21}B & a_{22}B & \dots & a_{2s}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1}B & a_{s2}B & \dots & a_{ss}B \end{pmatrix}.$$

We need the following well-known properties of eigenvalues and eigenvectors of tensor products of matrices.

LEMMA 3.3. *Let A be a square matrix of order s with eigenvalues $\alpha_1, \dots, \alpha_s$ and eigenvectors e_1, \dots, e_s and let B be a square matrix of order t with eigenvalues β_1, \dots, β_t and eigenvectors f_1, \dots, f_t . Then the eigenvalues of the matrix $A \otimes B$ are equal to $\alpha_i \beta_j, i = 1, \dots, s, j = 1, \dots, t$, and their corresponding eigenvectors are $e_i \otimes f_j$.*

We also need the following result.

LEMMA 3.4. *For every two integers $0 < Y < X$ and every integer $L > 0$, there is an integer g such that L divides the binomial coefficient $\binom{gX}{gY}$.*

Proof. If S, T are two positive integers and R is their sum, then for every prime p the maximum power of p that divides $\binom{R}{S} = \frac{R!}{S!T!}$ is p^h , where $h = \sum_{i \geq 1} (\lfloor R/p^i \rfloor - \lfloor S/p^i \rfloor - \lfloor T/p^i \rfloor)$. For each i , the i th term in the sum above is either 0 or 1, and it is 1 if and only if there is a carry in the i th rightmost digit when S and T are represented in base p and are being added to get R . Therefore in order to prove the lemma it suffices to show that for every finite collection of primes P and for every positive integer u the following holds. There is an integer g such that for $Z = X - Y$, and for every $p \in P$, when gY and gZ are being added in base p there is a carry in at least u places. We proceed with a proof of this fact.

Fix a prime $p \in P$ and consider the representations of Y and Z in base p . If the rightmost nonzero digit in both of them appears at the same place, then there is some $g_1 > 0$ such that the rightmost nonzero digit of g_1Y is $p - 1$, and as the digit of g_1Z in the same place is nonzero as well, there will be a carry in this position while the two numbers will be added. Otherwise assume, without loss of generality, that the rightmost nonzero digit of Y is to the right of the rightmost nonzero digit of Z . Choose $g_1 > 0$ such that the rightmost nonzero digit of g_1Z is $p - 1$. If the digit of g_1Y in this position is nonzero, then when adding g_1Y and g_1Z there will be a carry here. Otherwise, by defining $g'_1 = g_1(1 + p^s)$, where s is chosen so that the rightmost nonzero digit of g_1Yp^s is at the same place as the rightmost nonzero digit of g_1Z , we conclude that when adding g'_1Y and g'_1Z we have a carry in this place. We have thus shown that in all cases there is some positive g_1 such that when adding g_1Y and g_1Z there is a carry in at least one position. To get a carry in at least u positions we now take a sufficiently large integer m and define $g_p = g_1(1 + p^m + p^{2m} + \dots + p^{(u-1)m})$. If m is sufficiently large (as a function of Y, Z, g_1), then the representation of g_pY in base p consists of u pairwise disjoint blocks separated by zeros, where each block contains the representation of g_1Y . As the same description applies to gZ as well, we conclude that indeed when adding g_pY and g_pZ in base p there will be a carry in at least u places.

It remains to combine all the different numbers g_p and obtain the required g . For each $p \in P$, let p^{t_p} be a power of p satisfying $p^{t_p} > \max\{g_pY, g_pZ\}$. Note that if $g \equiv g_p \pmod{p^{t_p}}$, then the right part of the representation of gY in base p is identical to the representation of g_pY in base p , and the same holds for gZ . By the Chinese remainder theorem there is an integer g satisfying $g \equiv g_p \pmod{p^{t_p}}$ for all $p \in P$. It follows that p^u divides $\binom{gX}{gY}$ for all $p \in P$, completing the proof. \square

Having finished all necessary preparations, we are now ready to complete the proof of Theorem 3.1.

Proof of Theorem 3.1. Let $H = H(m, b)$ with $\frac{m-2b}{m} = \eta$ and adjacency matrix $A(H)$. By the above discussion this is a $d = \binom{m}{b}$ regular graph on 2^m vertices, and the smallest eigenvalue of $A(H)$ is equal to $\lambda = \frac{m-2b}{m} \binom{m}{b} = \eta d$. Choose an appropriate m such that $\frac{1-2a-\eta}{2a} \binom{m}{b}$ is an even, nonnegative integer. This is always possible, by Lemma 3.4, since a and η are rational numbers and $a \leq (1 - \eta)/2$. Pick H_1 to be any d_1 regular graph on $n_1 = \frac{1-2a-\eta}{2a} dd_1 + 1$ vertices such that if $\mu_1 \leq \mu_2 \leq \dots \leq \mu_{n_1-1} \leq \mu_{n_1} = d_1$ are all eigenvalues of $A(H_1)$ and $\mu = \max\{|\mu_1|, |\mu_{n_1-1}|\}$, then $\mu \leq \frac{2a-1}{2a} d_1$. There are several known constructions of such expander graphs. In particular, a random d_1 regular graph on n_1 vertices with high probability satisfies that $\mu = O(\sqrt{d_1})$ (see, e.g., [Fri91] and [FKS89]). By taking d_1 and n_1 sufficiently

large we obtain a graph with the desired properties. Denote by I the identity matrix of order 2^m and by K_{n_1} a complete graph on n_1 vertices. It is easy to see that the eigenvalues of the adjacency matrix $A(K_{n_1})$ are equal to $n_1 - 1$ and -1 (the latter with multiplicity $n_1 - 1$) and the corresponding eigenvectors are the all one vector $\mathbf{1}_{n_1}$ and any maximal set of independent vectors in R^{n_1} whose sums of coordinates equal zero. This implies that the adjacency matrices of H_1 and K_{n_1} have a common basis of eigenvectors.

Finally, let G be a graph on $n = 2^m n_1$ vertices with the following adjacency matrix:

$$A(G) = A(H) \otimes A(H_1) + I \otimes A(K_{n_1}).$$

In other words, G is obtained by replacing each vertex of H by a copy of K_{n_1} and replacing each edge of H by (the double cover of) a copy of H_1 . By the definition, $A(G)$ is a symmetric matrix with all entries equal to either 1 or 0 and for every row of $A(G)$ the sum of its entries is equal to $d' = dd_1 + (n_1 - 1)$. Therefore it is an adjacency matrix of a d' regular graph. The matrices $A(H)$ and I and also the matrices $A(H_1)$ and K_{n_1} have a common basis of eigenvectors. Thus by Lemma 3.3 we obtain that the same is true for $A(G)$, $A(H) \otimes A(H_1)$, and $I \otimes A(K_{n_1})$. Next, we need to compute the smallest eigenvalue of $A(G)$. By Lemma 3.3 it is easy to see that the only two possibilities for its value are $\lambda d_1 + (n_1 - 1)$ or $-\mu d - 1$. Since $-1 < \eta < 0$ and $d = \binom{m}{b}$ is large enough, an easy computation shows that

$$\begin{aligned} \lambda d_1 + (n_1 - 1) &= \eta dd_1 + \frac{1 - 2a - \eta}{2a} dd_1 = \frac{(1 - 2a)(1 - \eta)}{2a} dd_1 \\ &< \frac{(1 - 2a)}{2a} dd_1 - 1 < -\mu d - 1, \end{aligned}$$

where in the penultimate inequality we used the fact that $\frac{1-2a}{2a}\eta dd_1 > 1$ for all sufficiently large d_1 , and in the last inequality we used that $\mu \leq \frac{2a-1}{2a}d_1$. Therefore we conclude that the smallest eigenvalue of $A(G)$ is $\lambda d_1 + (n_1 - 1)$. Furthermore, by Lemma 3.3 and the properties of $H(m, b)$ this eigenvalue has multiplicity m and its corresponding eigenvectors are equal to $z_1 = y_1 \otimes \mathbf{1}_{n_1}, \dots, z_m = y_m \otimes \mathbf{1}_{n_1}$.

Clearly $z_i = (z_{i1}, \dots, z_{in})$ is a vector with ± 1 coordinates. Thus the coordinates of z_i correspond to a cut in G of size equal to

$$\begin{aligned} \sum_{k < j} a_{kj}(G) \frac{1 - z_{ik} z_{ij}}{2} &= \frac{1}{2} |E(G)| - \frac{1}{4} z_i^t A(G) z_i = \frac{1}{4} d' n - \frac{1}{4} (\lambda d_1 + (n_1 - 1)) \|z_i\|^2 \\ &= \frac{1}{4} \left(dd_1 + \frac{1 - 2a - \eta}{2a} dd_1 \right) n - \frac{1}{4} \left(\eta dd_1 + \frac{1 - 2a - \eta}{2a} dd_1 \right) n \\ &= \frac{1}{2} \frac{1 - \eta}{2} dd_1 n = a \frac{1}{2} d' n = a |E(G)|, \end{aligned}$$

where here we used the fact that $d' = dd_1 + n_1 - 1 = \frac{1-\eta}{2a} dd_1$. Thus the size of a maximum cut in G is equal to the optimal value of the semidefinite program (see Lemma 3.2). On the other hand, let B be the $2^m \times m$ matrix whose rows are the vectors w_i and thus its columns are equal to $\frac{1}{\sqrt{m}} y_i$. Denote by u_1, \dots, u_n the rows of the matrix $B \otimes \mathbf{1}_{n_1}$. By definition, $\|u_i\|^2 = 1$ and the columns of this matrix are the eigenvectors $\frac{1}{\sqrt{m}} y_i \otimes \mathbf{1}_{n_1} = \frac{1}{\sqrt{m}} z_i$ of $A(G)$ corresponding to its smallest eigenvalue

$\lambda d_1 + (n_1 - 1)$. Therefore by Lemma 3.2 it follows that

$$\begin{aligned} \max_{\|v_l\|^2=1} \sum_{i<j} a_{ij}(G) \frac{1 - v_i^t v_j}{2} &= \frac{1}{2} |E(G)| - \frac{1}{4} (\lambda d_1 + (n_1 - 1)) n = a |E(G)| \\ &= \sum_{i<j} a_{ij} \frac{1 - u_i^t u_j}{2} . \end{aligned}$$

To finish the proof of the theorem note that by definition each u_i is equal to one of the vectors w_k . In addition, if i and j are adjacent vertices in G , then u_i, u_j are equal either to the same vector w_k , and then $u_i^t u_j = 1$, or are equal to w_k, w_l which correspond to adjacent vertices in H , and in that case $u_i^t u_j = \eta$. \square

4. Approximation algorithms for the MAX NAE- $\{3\}$ -SAT problem.

An instance of the MAX 2-XOR (or MAX NAE- $\{2\}$ -SAT) problem in the variables x_1, x_2, \dots, x_n is composed of a (weighted) collection of pairs of the form $\langle z_1, z_2 \rangle$, where the z_i 's are literals. A clause $\langle z_1, z_2 \rangle$ is satisfied by a 0-1 assignment to the variables x_1, x_2, \dots, x_n if and only if $z_1 \neq z_2$ under this assignment. It is easy to see that instances of MAX CUT are just instances of MAX 2-XOR with no negations. The approximation algorithms of [GW95] and [Zwi99] are, in fact, approximation algorithms for MAX 2-XOR, not just for MAX CUT. The performance guarantees obtained by these algorithms on MAX 2-XOR instances are the same as those obtained on MAX CUT instances.

An instance of MAX NAE- $\{3\}$ -SAT is easily converted into an instance of MAX 2-XOR. Simply replace each triplet $\langle z_1, z_2, z_3 \rangle$ by the three pairs $\langle z_1, z_2 \rangle, \langle z_1, z_3 \rangle$, and $\langle z_2, z_3 \rangle$, giving each one of them a weight of $1/2$. It is easy to check that the total weight of the triplets/pairs satisfied by this transformation is unchanged. Thus, the algorithm of [GW95] is also an approximation algorithm for MAX NAE- $\{3\}$ -SAT with a performance ratio of at least $\alpha \simeq 0.87856$.

A better performance guarantee for the MAX NAE- $\{3\}$ -SAT problem can be obtained as follows. It is convenient to adopt the notation $x_{n+i} = \bar{x}_i$ for $1 \leq i \leq n$. If we let $w_{ijk} \geq 0$ be the weight attached to the triplet $\langle x_i, x_j, x_k \rangle$ in a MAX NAE- $\{3\}$ -SAT instance, then we can write the following semidefinite programming relaxation corresponding to the instance:

$$\begin{aligned} \max \quad & \sum_{i<j<k} w_{ijk} \frac{3 - v_i^t v_j - v_i^t v_k - v_j^t v_k}{4} \\ \text{such that} \quad & v_i^t v_i = 1, \quad v_i^t v_{n+i} = -1 \quad \text{for } 1 \leq i \leq n, \\ & v_i^t v_j + v_i^t v_k + v_j^t v_k \geq -1 \quad \text{for } 1 \leq i, j, k \leq 2n. \end{aligned}$$

If we round an optimal solution v_1, v_2, \dots, v_n of the above relaxation using a random hyperplane, then we still get a performance guarantee of only $\alpha \simeq 0.87856$. However, for *satisfiable* instances of MAX NAE- $\{3\}$ -SAT a performance guarantee of at least $\beta_1 = \frac{3}{2\pi} \arccos(-\frac{1}{3}) \simeq 0.91226$ is obtained (see [AKMR96], [Zwi98]). The performance ratio obtained is no better than β_1 if there exist unit vectors v_1, v_2, \dots, v_n such that $v_i^t v_j = v_i^t v_k = v_j^t v_k = -\frac{1}{3}$ for every triplet $\langle x_i, x_j, x_k \rangle$ of the instance with nonzero w_{ijk} . (Due to the constraint $v_i^t v_j + v_i^t v_k + v_j^t v_k \geq -1$, such a collection of vectors is automatically an optimal solution of the relaxation.) We show in the next sections that such solutions do exist.

Zwick [Zwi99] obtains a performance guarantee of at least $\beta \simeq 0.908718$ for general, not necessarily satisfiable, instances of the problem by constructing a sequence of vectors w_1, w_2, \dots, w_n such that $w_i^t w_j = c(v_i^t v_j)$, for every $i \neq j$, where $c \simeq 0.9789916$,

and then rounding w_1, w_2, \dots, w_n , and not v_1, v_2, \dots, v_n , using a random hyperplane. More specifically, the constants $\beta \simeq 0.908718$, $c \simeq 0.9789916$, and $\eta \simeq -0.74335866$ are the solutions of the three equations

$$c^2 \left(\eta^2 + \frac{4}{\beta^2 \pi^2} \right) = 1, \quad \frac{2 \arccos(c\eta) + \arccos(c)}{\pi} = \beta(1 - \eta), \quad \frac{3}{2\pi} \arccos\left(-\frac{c}{3}\right) = \beta.$$

It is further shown in [Zwi99] that the performance ratio achieved by this algorithm on a given instance is no better than β if the optimal value of the relaxation is equal to the optimal value of the instance (no integrality gap) and if for every triplet $\langle x_i, x_j, x_k \rangle$ that appears in the instance either $v_i^t v_j = v_i^t v_k = v_j^t v_k = -\frac{1}{3}$ or two of the inner products $v_i^t v_j$, $v_i^t v_k$, and $v_j^t v_k$ are η and the third is 1. Furthermore, a fraction of about $r \simeq 0.278797$ of the triplets should be of the second type; otherwise an improved ratio may be obtained by varying c . (We omit the exact equation defining r .) We are again able to show that such instances do exist, and hence the analysis is again tight.

5. Worst case instances for the MAX NAE- $\{3\}$ -SAT algorithms. Let $\mathcal{H} = (V, E)$ be a 3-uniform hypergraph. (Every $e \in E$ satisfies $e \subseteq V$ and $|e| = 3$.) A cut of \mathcal{H} is again a partition of V into two sets S and $V - S$. A cut $(S, V - S)$ is said to cut a hyperedge $e \in E$ if and only if $0 < |e \cap S| < 3$. A maximum cut of \mathcal{H} is a cut that cuts the largest number of edges. It is easy to see that the problem of finding a maximum cut of \mathcal{H} corresponds to a MAX NAE- $\{3\}$ -SAT instance with no negations. We show that the analyses of the MAX NAE- $\{3\}$ -SAT algorithms described in the previous section are tight even on such instances.

A hypergraph $\mathcal{H} = (V, E)$ has a cut of size $|E|$ if and only if it is 2-colorable. The MAX NAE- $\{3\}$ -SAT instance corresponding to it is then satisfiable. The following theorem shows that the bound of $\beta_1 = \frac{3}{2\pi} \arccos(-\frac{1}{3}) \simeq 0.91226$ on the performance ratio achieved by the MAX NAE- $\{3\}$ -SAT approximation algorithm described in the previous section on satisfiable instances is tight.

THEOREM 5.1. *For infinitely many values of n , there exists a 2-colorable 3-uniform hypergraph $\mathcal{H} = (V, E)$, such that $V = \{1, 2, \dots, n\}$, and a sequence of unit vectors w_1, w_2, \dots, w_n such that $w_i^t w_j = w_i^t w_k = w_j^t w_k = -\frac{1}{3}$ for every $\{i, j, k\} \in E$.*

Proof. It is easy to construct such an example for $n = 4$. Simply let E be composed of all subsets of $V = \{1, 2, 3, 4\}$ of size 3. It is easy to check that $S = \{1, 2\}$ is a cut that cuts all the edges. Let w_1, w_2, w_3 , and w_4 be four unit vectors such that $w_i^t w_j = -\frac{1}{3}$ for every $1 \leq i < j \leq 4$. This can be done, for example, by taking $w_1 = \frac{1}{\sqrt{3}}(1, 1, 1)^t$, $w_2 = \frac{1}{\sqrt{3}}(1, -1, -1)^t$, $w_3 = \frac{1}{\sqrt{3}}(-1, 1, -1)^t$, and $w_4 = \frac{1}{\sqrt{3}}(-1, -1, 1)^t$.

This example is a special case of the following more general construction which supplies an infinite family of satisfiable instances of a MAX NAE- $\{3\}$ -SAT problem for which the analysis from [Zwi98] is tight. Let $H = H(m, b)$ with $b = 2m/3$. The vertex set $\{v_1, v_2, \dots\}$ of H consists of all ± 1 vectors of length m , and two vectors are adjacent if the number of coordinates in which they differ is equal to $2m/3$. Let $w_i = \frac{1}{\sqrt{m}} v_i$ for all i ; thus $\|w_i\|^2 = 1$ and $w_i^t w_j = \frac{m-2b}{m} = -1/3$ for any pair of adjacent vertices in H . Let \mathcal{H} be the 3-uniform hypergraph whose edges are triples of the vertices in H that form a cycle of length 3. By definition, it is easy to see that three vectors which form a triangle in the graph H cannot have the same first coordinate. By partitioning vertices into two parts according to their first coordinate, we therefore obtain a 2-coloring of \mathcal{H} , as required. \square

We next show that the analysis of the performance of the MAX NAE- $\{3\}$ -SAT algorithm of Zwick [Zwi99] on general, not necessarily satisfiable, instances is also

tight. By the discussion in the previous section, this follows from the following theorem. Given a 3-uniform hypergraph \mathcal{H} , we let $OPT(\mathcal{H})$ be the size of the maximum cut of \mathcal{H} , and we let $z^*(\mathcal{H})$ be the optimal value of the semidefinite programming relaxation of the corresponding MAX NAE- $\{3\}$ -SAT instance.

THEOREM 5.2. *Let $-1 < \eta < -1/2$ be a rational number and let $0 < r < 1$ and $\epsilon > 0$. Then, for infinitely many values of $|U|$ there exists a 3-uniform hypergraph $\mathcal{H} = (U, E)$, where $U = \{u_1, u_2, \dots\}$ and $E = E_1 \cup E_2$, and unit vectors w_{u_1}, w_{u_2}, \dots such that for every $\{u_i, u_j, u_k\} \in E_1$, we have $w_{u_i}^t w_{u_j} = w_{u_i}^t w_{u_k} = w_{u_j}^t w_{u_k} = -\frac{1}{3}$, for every $\{u_i, u_j, u_k\} \in E_2$ exactly two of the inner products $w_{u_i}^t w_{u_j}$, $w_{u_i}^t w_{u_k}$, $w_{u_j}^t w_{u_k}$ are η and the third is 1, and such that*

$$OPT(\mathcal{H}) = z^*(\mathcal{H}) = \sum_{\{i,j,k\} \in E} \frac{3 - w_{u_i}^t w_{u_j} - w_{u_i}^t w_{u_k} - w_{u_j}^t w_{u_k}}{4}.$$

In addition $|E_2|$ is bounded by $r|E| \leq |E_2| \leq (r + \epsilon)|E|$.

Proof. The hypergraph $\mathcal{H} = (U, E)$ that we construct is the union of two hypergraphs $\mathcal{H}_1 = (U, E_1)$ and $\mathcal{H}_2 = (U, E_2)$, that is, $E = E_1 \cup E_2$. We begin with the description of \mathcal{H}_1 .

Let m and n be (large) integers. (Their values are specified at the end of the proof.) Let H_1 be the graph $H_1 = H(m, b)$ with $b = 2m/3$ (we assume that m is divisible by 3) and let I be an identity matrix of order n . Consider a graph G_1 with adjacency matrix $A(G_1) = A(H_1) \otimes I$. Clearly G_1 is just a disjoint union of n copies of H_1 . The vertex set $U = \{u_1, u_2, \dots\}$ of this graph consists of all pairs $\{(v, t) | v \in \{-1, 1\}^m, 1 \leq t \leq n\}$, and two vertices (v, t) and (v', t') are adjacent if and only if $t = t'$ and v and v' differ in exactly $2m/3$ coordinates. Let $w_u = \frac{1}{\sqrt{m}}v$ for all $u = (v, t)$; thus $\|w_u\|^2 = 1$ and $w_{u_i}^t w_{u_j} = \frac{m-2b}{m} = -1/3$ for any pair of adjacent vertices in G_1 . Let \mathcal{H}_1 be a 3-uniform hypergraph, whose edges are the triples of the vertices in G_1 which form a cycle of length 3. It is easy to see that the number of edges in \mathcal{H}_1 is equal to $\frac{1}{6}n2^m \binom{m}{2m/3} \binom{2m/3}{m/3}$. Let A be a subset of U containing all vertices (v, t) with first coordinate of v_i equal to one and let $B = U - A$. It follows easily from the definition that the three vertices of a 3-cycle in G_1 cannot all have the same first coordinate. Thus any 3-cycle in G_1 will intersect both A and B . Therefore we obtain a cut in the hypergraph \mathcal{H}_1 whose size is equal to the total number of edges of \mathcal{H}_1 . Finally, since $w_{u_i}^t w_{u_j} = w_{u_i}^t w_{u_k} = w_{u_j}^t w_{u_k} = -1/3$ for any edge in \mathcal{H}_1 and the value of the semidefinite relaxation $z^*(\mathcal{H}_1)$ is always bounded by $|E(\mathcal{H}_1)|$ we conclude that

$$\begin{aligned} \sum_{\{u_i, u_j, u_k\} \in E(\mathcal{H}_1)} \frac{3 - w_{u_i}^t w_{u_j} - w_{u_i}^t w_{u_k} - w_{u_j}^t w_{u_k}}{4} &= |E(\mathcal{H}_1)| = OPT(\mathcal{H}_1) = z^*(\mathcal{H}_1) \\ &= \frac{1}{6}n2^m \binom{m}{2m/3} \binom{2m/3}{m/3}. \end{aligned}$$

The construction of \mathcal{H}_2 is more involved than that of \mathcal{H}_1 . We start by constructing an auxiliary multigraph G_2 . Let H_2 be the graph $H_2 = H(m, b)$ with $b = \frac{1-\eta}{2}m$. (η is given at the statement of the theorem and can be made arbitrarily close to $-0.74335866\dots$) Let K_n be a complete graph on n vertices. The graph H_2 is d regular with $d = \binom{m}{(1-\eta)m/2}$, and, by the discussion in section 3, the smallest eigenvalue of its adjacency matrix $A(H_2)$ is $\lambda = \frac{m-2b}{m}d = \eta d$. Let G_2 be a multigraph with adjacency matrix equal to $A(G_2) = (A(H_2) + dI/2) \otimes A(K_n)$, where I is an identity matrix of

order 2^m . The vertex set of G_2 is again $U = \{(v, t) | v \in \{-1, 1\}^m, 1 \leq t \leq n\}$, and two vertices (v, t) and (v', t') are connected by a unique edge if $t \neq t'$ and v and v' differ in exactly $\frac{1-\eta}{2}m$ coordinates or they are connected by $d/2$ parallel edges if $v = v'$ and $t \neq t'$. By definition G_2 is a $(3d(n-1)/2)$ regular multigraph and by Lemma 3.3 its smallest eigenvalue is equal to $(\lambda+d/2)(n-1) = (\eta+1/2)d(n-1) < -3d/2 < 0$, where here we used the fact that n is sufficiently large. Let $w_u = \frac{1}{\sqrt{m}}v$ for all vertices (v, t) be as before; thus $\|w_u\|^2 = 1$ and $w_{u_i}^t w_{u_j} = \frac{m-2b}{m} = \eta$ or 1 for any pair of adjacent vertices of G_2 . In addition $w_{u_i}^t w_{u_j} = 1$ if and only if $u_i = (v, t)$ and $u_j = (v, s)$ with $t \neq s$. Let $B = (b_{ij})$ be the $2^m n \times m$ matrix whose rows are equal to the vectors $w_u, u \in U$. Note that the elements in B are $\pm \frac{1}{\sqrt{m}}$. As in the proof of Theorem 3.1, we can see that the columns of B are eigenvectors of $A(G_2)$ that correspond to the smallest eigenvalue of $A(G_2)$. Let $OPT(G_2)$ be the size of the MAX CUT in G_2 and let $z^*(G_2)$ be the value of the semidefinite programming relaxation. Then by Lemma 3.2 we obtain that

$$\begin{aligned} OPT(G_2) &= z^*(G_2) = \sum_{i < j} a_{ij}(G_2) \frac{1 - w_{u_i}^t w_{u_j}}{2} \\ &= \frac{1}{2} |E(G_2)| - \frac{1}{4} (\eta + 1/2) d(n-1) |V(G_2)|. \end{aligned}$$

The coordinates of the first column of B produce the cut (A, B) (same as for \mathcal{H}_1) and its size is equal to $OPT(G_2)$, since the first column of B is an eigenvector of the smallest eigenvalue of $A(G_2)$.

Now we are ready to construct \mathcal{H}_2 . Let \mathcal{H}_2 be the 3-uniform hypergraph on the vertex set $U = V(G_2)$, whose edges are the following triples of the vertices of G_2 ; $\{u_i, u_j, u_k\}$ belongs to $E(\mathcal{H}_2)$ if and only if $u_i = (v, t)$, $u_j = (v, t')$ and $u_k = (v'', t'')$ such that $t \neq t' \neq t''$ and v and v'' differ in exactly $(1-\eta)m/2$ coordinates. Note that by definition, the number of edges in \mathcal{H}_2 is equal to $\frac{1}{2}n(n-1)(n-2)2^m \binom{m}{(1-\eta)m/2}$ and they form cycles of length 3 in G_2 . In addition every edge of G_2 connecting $u_i = (v, t)$ and $u_k = (v'', t'')$ (as above) is contained in exactly $2(n-2)$ edges of \mathcal{H}_2 and every pair of vertices $u_i = (v, t)$ and $u_j = (v, t')$ (as above) is contained in exactly $d(n-2)$ edges of \mathcal{H}_2 . Since in the multigraph G_2 between the vertices $u_i = (v, t)$ and $u_j = (v, t')$ we have $d/2$ parallel edges, we can distribute them equally between all 3-cycles which correspond to the edges of \mathcal{H}_2 containing this pair of vertices. By doing this we obtain that every edge in the multigraph G_2 is contained in exactly $2(n-2)$ edges of \mathcal{H}_2 . In this case the size of the MAX CUT in \mathcal{H}_2 is closely related to the size of the MAX CUT of G_2 . Note that for any partition of the vertices $(X, U-X)$, the number of edges of \mathcal{H}_2 which crosses this cut is exactly $n-2$ times the number of edges of G_2 with the same property. Indeed, any edge from G_2 which connects X with $U-X$ is contained in $2(n-2)$ triples from \mathcal{H}_2 . All of them also cross this cut, but every such triple we counted twice, since it contributes two edges of G_2 to the cut. Therefore we can conclude that the value of MAX CUT of \mathcal{H}_2 is equal to $OPT(\mathcal{H}_2) = (n-2)OPT(G_2)$ and this value is obtained on the cut (A, B) , the same one as for the graph G_2 . This, together with the above discussion implies that

$$\begin{aligned} OPT(\mathcal{H}_2) &\leq z^*(\mathcal{H}_2) \leq \max \sum_{\{u_i, u_j, u_k\} \in E(\mathcal{H}_2)} \frac{3 - y_{u_i}^t y_{u_j} - y_{u_i}^t y_{u_k} - y_{u_j}^t y_{u_k}}{4} \\ &= \frac{1}{2} \max \sum_{\{u_i, u_j, u_k\} \in E(\mathcal{H}_2)} \left[\frac{1 - y_{u_i}^t y_{u_j}}{2} + \frac{1 - y_{u_i}^t y_{u_k}}{2} + \frac{1 - y_{u_j}^t y_{u_k}}{2} \right] \end{aligned}$$

$$\begin{aligned}
&\leq (n-2) \max_{\{u_i, u_j\} \in E(G_2)} \sum \frac{1 - y_{u_i}^t y_{u_j}^t}{2} = (n-2) \sum_{i < j} a_{ij}(G_2) \frac{1 - w_{u_i}^t w_{u_j}^t}{2} \\
&= (n-2) z^*(G_2) = (n-2) OPT(G_2) = OPT(\mathcal{H}_2).
\end{aligned}$$

Thus,

$$OPT(\mathcal{H}_2) = z^*(\mathcal{H}_2) = \sum_{\{u_i, u_j, u_k\} \in E(\mathcal{H}_2)} \frac{3 - w_{u_i}^t w_{u_j}^t - w_{u_i}^t w_{u_k}^t - w_{u_j}^t w_{u_k}^t}{4}.$$

Also, we know that for every edge $\{u_i, u_j, u_k\}$ of \mathcal{H}_2 , two of the inner products $w_{u_i}^t w_{u_j}^t$, $w_{u_i}^t w_{u_k}^t$, and $w_{u_j}^t w_{u_k}^t$ are η and the third is 1.

Finally let \mathcal{H} be the 3-uniform hypergraph with the same vertex set U and with edge set $E(\mathcal{H}_1) \cup E(\mathcal{H}_2)$. We clearly have $OPT(\mathcal{H}) \leq z^*(\mathcal{H}) \leq z^*(\mathcal{H}_1) + z^*(\mathcal{H}_2) = OPT(\mathcal{H}_1) + OPT(\mathcal{H}_2)$. On the other hand, (A, B) is a MAX CUT of both \mathcal{H}_1 and \mathcal{H}_2 ; thus it is also a cut of \mathcal{H} of size $OPT(\mathcal{H}_1) + OPT(\mathcal{H}_2)$. As the same vectors w_{u_i} were used for the two hypergraphs, we get that

$$\begin{aligned}
\sum_{\{u_i, u_j, u_k\} \in E(\mathcal{H})} \frac{3 - w_{u_i}^t w_{u_j}^t - w_{u_i}^t w_{u_k}^t - w_{u_j}^t w_{u_k}^t}{4} &= OPT(\mathcal{H}_1) + OPT(\mathcal{H}_2) \\
&= OPT(\mathcal{H}) = z^*(\mathcal{H}).
\end{aligned}$$

In addition, for every edge $\{u_i, u_j, u_k\} \in E(\mathcal{H})$, either two of the inner products $w_{u_i}^t w_{u_j}^t$, $w_{u_i}^t w_{u_k}^t$, and $w_{u_j}^t w_{u_k}^t$ are η and the third is 1 (if $\{u_i, u_j, u_k\} \in E(\mathcal{H}_2)$) or $w_{u_i}^t w_{u_j}^t = w_{u_i}^t w_{u_k}^t = w_{u_j}^t w_{u_k}^t = -\frac{1}{3}$ (if $\{u_i, u_j, u_k\} \in E(\mathcal{H}_1)$). Finally, recall that $|E(\mathcal{H}_1)| = \frac{1}{6} n 2^m \binom{m}{2m/3} \binom{2m/3}{m/3}$ and that $|E(\mathcal{H}_2)| = \frac{1}{2} n (n-1)(n-2) 2^m \binom{m}{(1-\eta)m/2}$, where m and n are (large) parameters that we are free to choose. By choosing appropriate values of m and n , and using the fact that

$$\binom{m}{2m/3} \binom{2m/3}{m/3} > 2^m > \binom{m}{(1-\eta)m/2}$$

for all sufficiently large m , it follows that we can control the proportion of the edges of the second type, and make it arbitrarily close to r , as required. \square

6. Concluding remarks. We have shown that lower bounds on the performance guarantees of semidefinite programming based approximation algorithms for the MAX CUT, MAX 2-XOR, and MAX NAE- $\{3\}$ -SAT problems obtained using local ratio arguments are indeed tight.

Furthermore, our constructions show that the analyses of these algorithms are tight even if arbitrary collections of valid constraints are added to the semidefinite programming relaxations of these problems. Let $a_{ij}, 1 \leq i < j \leq n$, and b be real numbers. A constraint

$$\sum_{i < j} a_{ij} (v_i^t v_j^t) \geq b$$

is said to be *valid* if it is satisfied whenever each v_i is an integer in $\{-1, 1\}$. Feige and Goemans [FG95] and Goemans and Williamson [GW95] proposed adding valid constraints to the semidefinite relaxations in the hope of narrowing the gap between the optimal value of the semidefinite program and the weight of the optimal solution.

As all the coordinates of the vectors u_1, u_2, \dots of section 3 and of the vectors w_1, w_2, \dots of section 5 are equal to $\pm 1/\sqrt{m}$, it is not difficult to see that they satisfy any valid constraint. Thus the proofs of Theorems 3.1, 5.1, and 5.2 show that the addition of any collection of valid constraints does not improve the performance ratio of the abovementioned approximation algorithms for the MAX CUT, MAX 2-XOR, and MAX NAE- $\{3\}$ -SAT problems.

It is shown in [KZ97] that the $7/8$ lower bound on the performance ratio of the MAX 3-SAT approximation algorithm, obtained again using a local ratio argument, is also tight. Does local analysis always produce tight results? We see no reason why this should always be the case. It would be interesting to find natural approximation algorithms for interesting constraint satisfaction problems for which local analysis is not tight. It would also be interesting to know whether the local analyses of the approximation algorithms of Feige and Goemans [FG95] (see also [Zwi00]) for the MAX 2-SAT and MAX DI-CUT problems are tight. This seems, however, to require some additional techniques.

Acknowledgments. We would like to thank two anonymous referees for many helpful comments.

REFERENCES

- [AKMR96] N. ALON, P. KELSEN, S. MAHAGAN, AND H. RAMESH, *Approximate hypergraph coloring*, Nordic J. Comput., 3 (1996), pp. 425–439.
- [AS00] N. ALON AND B. SUDAKOV, *Bipartite subgraphs and the smallest eigenvalue*, Combin. Probab. Comput., 9 (2000), pp. 1–12.
- [CHLL97] G. COHEN, I. HONKALA, S. LITSYN, AND A. LOBSTEIN, *Covering Codes*, North-Holland, Amsterdam, 1997.
- [FG95] U. FEIGE AND M.X. GOEMANS, *Approximating the value of two prover proof systems, with applications to MAX-2SAT and MAX-DICUT*, in Proceedings of the 3rd Israel Symposium on Theory and Computing Systems, Tel Aviv, Israel, 1995, pp. 182–189.
- [FKS89] J. FRIEDMAN, J. KAHN, AND E. SZEMERÉDI, *On the second eigenvalue in random regular graphs*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, 1989, pp. 587–598.
- [Fri91] J. FRIEDMAN, *On the second eigenvalue and random walks in random d -regular graphs*, Combinatorica, 11 (1991), pp. 331–362.
- [GW95] M.X. GOEMANS AND D.P. WILLIAMSON, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42 (1995), pp. 1115–1145.
- [Hås97] J. HASTAD, *Some optimal inapproximability results*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, El Paso, TX, 1997, pp. 1–10.
- [Kar99] H. KARLOFF, *How good is the Goemans–Williamson MAX CUT algorithm?* SIAM J. Comput., 29 (1999), pp. 336–350.
- [KZ97] H. KARLOFF AND U. ZWICK, *A $7/8$ -approximation algorithm for MAX 3SAT?*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, FL, 1997, pp. 406–415.
- [Lov93] L. LOVÁSZ, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, 1993.
- [TSSW96] L. TREVISAN, G.B. SORKIN, M. SUDAN, AND D.P. WILLIAMSON, *Gadgets, approximation, and linear programming (extended abstract)*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, VT, 1996, pp. 617–626.
- [Zwi98] U. ZWICK, *Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint*, in Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, ACM, New York, 1998, pp. 201–210.

- [Zwi99] U. ZWICK, *Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, Atlanta, GA, 1999, pp. 679–687.
- [Zwi00] U. ZWICK, *Analyzing the MAX 2-SAT and MAX DI-CUT approximation algorithms of Feige and Goemans*, submitted.

NEIGHBORHOOD PRESERVING HASHING AND APPROXIMATE QUERIES*

DANNY DOLEV[†], YUVAL HARARI[†], NATHAN LINIAL[†], NOAM NISAN[†], AND
MICHAL PARNAS[‡]

Abstract. Let $D \subseteq \Sigma^n$ be a dictionary. We look for efficient data structures and algorithms to solve the following *approximate query* problem: Given a query $u \in \Sigma^n$ list all words $v \in D$ that are close to u in Hamming distance.

The problem reduces to the following combinatorial problem: Hash the vertices of the n -dimensional hypercube into buckets so that (1) the c -neighborhood of each vertex is mapped into at most k buckets and (2) no bucket is too large.

Lower and upper bounds are given for the tradeoff between k and the size of the largest bucket. These results are used to derive bounds for the approximate query problem.

Key words. approximate query, hashing, isoperimetric inequality, error correcting code

AMS subject classifications. 68P20, 68P05, 68P10, 05B40, 05D99

PII. S089548019731809X

1. Introduction. Consider a text with words that all belong to a given dictionary D . Due to limited reliability, the words may contain errors. Our task is to provide for every word from the text a list of alternative similar words from the dictionary, while minimizing the search time and storage space. When the size of D is small enough, it may be feasible to search all of it for each such query (as done, for example, by algorithms for the problem of string matching with k -differences; see [LV1], [LV2]). However, when D is large, a more efficient approach is needed. This problem, called the *approximate query* problem, can arise in many different fields. The most obvious examples come from the design of efficient spellers and speech-recognizers. Variations of the problem arise in fields such as the analysis of DNA sequences and proteins in chemistry and biology.

Hashing is a powerful tool in handling many similar problems. However, as we shall see, the hash functions we require also need to *preserve locality*. The requirements of hashing and of preserving locality seem to contradict one another. Informally, the objective of hashing is to “scatter” the given data, while preserving locality means doing the opposite. Both hashing and locality preservation are used quite widely in theory and in practice, but the tradeoffs between these two requirements are not well understood. The combinatorial problems raised in this paper concern the extent to which hashing and preservation of locality can be satisfied together.

In the scenario under consideration, the dictionary D may change over time, and we desire a data structure that requires only minimal modifications upon changes in D . For concreteness, let $D \subseteq \{0, 1\}^n$. Our approach is to look for a mapping $h : \{0, 1\}^n \rightarrow \{1, \dots, B\}$, where B is the number of entries (buckets) in the hash table. Given a query $u \in \{0, 1\}^n$, we would like to retrieve all words in D that

*Received by the editors March 7, 1997; accepted for publication (in revised form) November 15, 2001; published electronically January 4, 2002.

<http://www.siam.org/journals/sidma/15-1/31809.html>

[†]Institute of Computer Science, The Hebrew University, Jerusalem, Israel (dolev@cs.huji.ac.il, nati@cs.huji.ac.il, noam@cs.huji.ac.il). The work of the first author was supported in part by the Yeshaya Horowitz Association. The work of the third author was supported in part by a grant from the Israeli Academy of Sciences.

[‡]The Academic College of Tel-Aviv-Yaffo, 4 Antokolsky st., Tel-Aviv, Israel (michalp@mta.ac.il).

are in the c -neighborhood of u (i.e., words of Hamming distance at most c from u). Thus, in order to minimize the search time, h should map each c -neighborhood into a small number, k , of buckets. This requirement reflects our desire for h to preserve locality. At the same time, we expect h to exhibit good hashing properties, in that each bucket should contain relatively few words. This requirement reduces the amount of redundant search among words not in the c -neighborhood of u . We provide both upper and lower bounds on the tradeoff between k , c , and the size of the largest bucket. These tradeoffs are used in section 2 to derive bounds on the complexity of the approximate query problem.

A framework for questions of this type was developed in [DHP], where the problem is called the *approximate query retrieval* problem (see also [H] and [P]). A similar problem called the *partial-match retrieval* problem is studied in [R]. Numerous interesting applications of such problems can be found in [SK].

We can now state formally our problem and results. The (c, k) -coloring problem is defined as follows:

Color the vertices of the n -dimensional cube C_n so that the c -neighborhood of each vertex is colored with at most k colors and such that the largest color class is as small as possible.

We call such a coloring a (c, k) -coloring and say that each c -neighborhood is k -colored. Our first results are lower bounds on the size of the largest color class. These bounds rely on isoperimetric inequalities for the cube C_n and are described in section 3. Specifically we prove the following theorems.

THEOREM 1.1. *In any $(1, k)$ -coloring of C_n there exists a color that appears at least $\sum_{i=0}^t \binom{n}{i}$ times, where $t = \frac{n+2}{k+1} - 1$.*

THEOREM 1.2. *In any (c, k) -coloring of C_n , there exists a color that appears at least $\sum_{i=0}^t \binom{n}{i}$ times, where $t = \frac{n-c\sqrt[k-1]{k-1}}{\sqrt[k-1]{k-1}+2}$.*

Some of the applications require coloring only some small subset of the cube. The following lower bound addresses this issue and can be found in section 4. Denote by $B_{n,r}$ the Hamming ball of radius r in C_n .

THEOREM 1.3. *In any $(1, k)$ -coloring of $B_{n,r}$, there exists a color that appears at least $\binom{n/k}{r}$ times.*

The upper bounds for the (c, k) -problem are described in section 5. Although we do not have a full answer, we show various colorings of the cube that match, or nearly match, our lower bounds. Some of these colorings are derived using error correcting codes (see [PW] or [MS] for relevant information on error correcting codes). Examples of the upper bounds obtained are described in Table 1.1.

We conclude by introducing a more general problem called the (c, k, s) -covering problem, which allows each word to be hashed to s entries in the hash table instead of to a single entry, using s hash functions instead of only one. Section 6 contains a combinatorial formalization of this problem, as well as possible directions for further research.

2. The approximate query problem. Let $D \subseteq \Sigma^n$ be a dictionary of words of length n over some alphabet Σ . We concentrate on the case where $\Sigma = \{0, 1\}$, as our combinatorial results directly translate to this case. The extensions to a larger alphabet size are straightforward. Our results apply to algorithms of the following type (see also [DHP]).

The data structure. The algorithm stores the dictionary D in a hash table using a hash function $h_D : \Sigma^n \rightarrow \{1 \dots B\}$, where B is the number of entries in the

TABLE 1.1
Upper bounds for the $(1, k)$ -coloring problem.

n	k	Largest color	Remarks
any	n	2	
any	$n - d + 1$	2^d	d is a constant
$2^r - 1$	$(n + 1)/2$	$n + 1$	uses Hamming codes
$4^r - 1$	$1 + n/3$	$O(n^3)$	uses nearly perfect codes
see	remarks	$\sum_{i=0}^{t+1} \binom{n}{i},$ $t = O(n/\sqrt{k})$	when an (n, t) -quasi-perfect code exists
any	any	$\sum_{i=0}^{t+d} \binom{n}{i},$ $t = O(n/k^{1/(d+1)})$	$d \leq t,$ uses a general (n, t, d) -code
any	2	$2 \binom{n}{n/2}$	

hash table. Each entry (bucket) in the hash table contains all the words in D that were hashed to it.

Answering queries. Define the c -neighborhood of $u \in \Sigma^n$ with respect to Σ^n to be

$$N_c(u) = \{v | v \in \Sigma^n, d(u, v) \leq c\},$$

where $d(u, v)$ is the Hamming distance of u and v . The c -neighborhood of $u \in \Sigma^n$ with respect to D is

$$N_c(u, D) = N_c(u) \cap D.$$

The algorithm is given queries $u \in \Sigma^n$ and should determine $N_c(u, D)$. The algorithm answers a query u by probing those buckets in the hash table that contain $N_c(u, D)$. Let S_u be the sequence of buckets searched by the algorithm as an answer to query u . The sequence S_u is called the *search sequence* of u .

Complexity measures. The total time to answer a query u depends on the length $|S_u|$ of the search sequence of u and the total number of words found in the buckets searched by S_u . We thus have two complexity measures:

1. The length $K_D = \max_u |S_u|$ of the longest search sequence used by the algorithm. Small search sequences are advantageous, since each access to a different bucket may result in a costly disk access.
2. The size M_D of the largest bucket created by the algorithm, i.e., the maximum number of words in D that were mapped by h_D to the same bucket. A small M_D reduces the number of irrelevant words read by the algorithm when answering a query u .

We first show that there exist dictionaries D , for which M_D grows exponentially in n/K_D , for any hash function h_D .

COROLLARY 2.1. *There exists a dictionary D such that for every hash function h_D , $M_D \geq 2^{\Omega(n/K_D)}$.*

Proof. Let $D = \Sigma^n$. Then any hash function h_D for which the length of the search sequences is bounded by K_D is a K_D -coloring of the cube C_n . The claim then follows from Theorems 1.1 and 1.2, using the Stirling approximation for the binomial coefficients. \square

Note that if c is a constant, then the c -neighborhood of any query is polynomial in size (even for the large dictionary D described in the proof). Yet if K_D is small, then there will be a bucket which contains many words that are irrelevant to a query whose search sequence includes this bucket. For example, if $D = \{0, 1\}^n$, then the 1-neighborhood of each query contains $n + 1$ words. However, if K_D is a constant, then Corollary 2.1 states that there exists a bucket that contains $2^{\Omega(n)}$ words.

The proof of Corollary 2.1 holds only for big dictionaries. Are there any good hash functions for small dictionaries? Notice that for a small enough random dictionary D , the size of the c -neighborhood of any query u with respect to D is $|N_c(u, D)| = O(1)$ with high probability. Thus it is possible to map each word of D to a separate bucket, and there exists a short search sequence for each query (although computing this search sequence may not be easy). We exhibit, however, small dictionaries with no good hash function h . That is, either the search sequence is long or there exists a large bucket.

COROLLARY 2.2. *There exists a dictionary D of size $|D| = O(n^r)$ such that for every hash function h_D , $M_D = \Omega(|D|/K_D^r)$.*

Proof. Let D be a dictionary that includes all the words in some Hamming ball of radius r . The claim follows from Theorem 1.3. \square

It is possible to define a number of natural complexity measures in terms of K_D and M_D . In [DHP] one such possible time complexity measure is defined

$$Time_A(h_D, u) = |S_u| + \sum_{i \in S_u} |B_i|,$$

where A is an algorithm that uses a hash function h_D , $u \in \Sigma^n$ is a query, and B_i is the i th bucket in the hash table. The efficiency of A in terms of its time complexity can then be measured as

$$T_A = \max_{D, u} \frac{TIME(h_D, u)}{|N_c(u, D)| + 1}.$$

Algorithm A is *time optimal* if $T_A = O(1)$; that is, the algorithm gives an answer in time linear in the answer size.

An algorithm for which the hash function h is fixed for any dictionary D , and the search sequence of a query u depends only on h and u , will be called *D -oblivious*. Such algorithms are of course preferred since they are easier to design, and can handle a dynamically changing dictionary D , without having to change the data structure with every change in D . For such algorithms we give tight lower bounds on the time complexity defined above. Denote by N_c the size of a c -neighborhood with respect to Σ^n . Thus for the Hamming distance $N_c = \sum_{i=0}^c \binom{n}{i} (|\Sigma| - 1)^i$. Note that usually $|N_c(u, D)| \ll N_c$. Let A be a D -oblivious algorithm that is *space optimal* (i.e., uses space $|D|$). Then [DHP] show that $T_A = \Omega(\sqrt{N_c})$. We improve their result and show the following corollary.

COROLLARY 2.3. *Let A be an optimal space D -oblivious algorithm. Then $T_A = \Omega(N_c/4^c)$.*

Proof. Since algorithm A is D -oblivious, it uses some fixed hash function h for all dictionaries D . If $K_A > N_c/4^c$, then there exists a query u for which $|S_u| > N_c/4^c$. Let D be any dictionary for which the c -neighborhood of u is empty; that is, $|N_c(u, D)| = 0$. The claim follows.

If $K_A \leq N_c/4^c$, then h is an $(N_c/4^c)$ -coloring of the cube. Thus by Theorem 1.1 and Theorem 1.2, there exists a color set of size $\Omega(N_c^2/4^c)$. Let D be a dictionary

that includes most of the words in this color set. The claim follows for any query u whose search sequence includes this set. \square

This bound is tight up to a factor of 4^c . Consider, for example, the D -oblivious algorithm that maps each word to a separate bucket. The search sequence of a query u will include the buckets of all its possible c -neighbors in Σ^n . Thus $|S_u| = |N_c(u)| = N_c$, and the time complexity is $N_c + |N_c(u, D)| = O(N_c)$.

3. Lower bounds on the size of the largest color set in any (c, k) -coloring of C_n . This section will provide lower bounds on the size of the largest color set in any (c, k) -coloring of the n -dimensional cube C_n . If $k = 1$, it is clear that the whole cube should be colored using one color. For the remainder of this and the next section assume therefore that $k \geq 2$. We now prove Theorems 1.1 and 1.2.

Recall that we assume that $\Sigma = \{0, 1\}$. Similar results can be obtained for a general alphabet Σ , using an extension of the isoperimetric inequality for larger alphabets. The details are omitted.

3.1. Proof of Theorem 1.1. Assume by contradiction that all color sets are smaller than the size stated in the theorem. The isoperimetric inequality provides a lower bound on the number of neighbors of a set of vertices in C_n , as a function of the cardinality of the set. Thus every color set S has at least $\beta|S|$ neighbors (where β is a decreasing function of $|S|$). The number of neighbors of all color sets can thus be bounded from below. On the other hand, the fact that the 1-neighborhood of each vertex is k -colored can be used to bound the number of neighbors of all color sets from above. An appropriate choice of parameters leads to a contradiction. We proceed with the detailed proof.

Denote by Γ_S the 1-neighborhood of a subset $S \subseteq C_n$, not including the vertices in S . That is,

$$\Gamma_S = \{v \notin S \mid \exists u \in S, d(u, v) = 1\}.$$

LEMMA 3.1 (isoperimetric inequality). *Let S be a subset of C_n of size $|S| = \sum_{i=0}^{r-1} \binom{n}{i} + m$, where $0 \leq m \leq \binom{n}{r}$. Then $|\Gamma_S| \geq \binom{n}{r} - m + m \cdot \binom{n}{r+1} / \binom{n}{r}$.*

Proof. See [B, pp. 122–129]. \square

Using this inequality, we can derive a lower bound on the ratio $|\Gamma_S|/|S|$ as a function of $|S|$.

LEMMA 3.2. *Let S be any set of size $|S| \leq \sum_{i=0}^r \binom{n}{i}$. Then $\frac{|\Gamma_S|}{|S|} \geq \binom{n}{r+1} / \sum_{i=0}^r \binom{n}{i}$.*

Proof. Let S be any set with size $|S| = \sum_{i=0}^{r-1} \binom{n}{i} + m$, where $0 \leq m \leq \binom{n}{r}$.

By Lemma 3.1, the claim holds for $m = \binom{n}{r}$, and so it suffices to show that the function

$$f(m) = \frac{\binom{n}{r} - m + m \cdot \binom{n}{r+1} / \binom{n}{r}}{\sum_{i=0}^{r-1} \binom{n}{i} + m}$$

is decreasing over $0 \leq m \leq \binom{n}{r}$. The inequality $f(m) \geq f(m + 1)$ reduces to showing that

$$\binom{n}{r+1} \cdot \sum_{i=0}^{r-1} \binom{n}{i} \leq \binom{n}{r} \cdot \sum_{i=0}^r \binom{n}{i}.$$

This inequality follows by comparing the i th term on the left with the $(i + 1)$ term on the right. \square

LEMMA 3.3.

- $\binom{n}{\alpha n + 1} = \frac{1-\alpha}{1/n+\alpha} \binom{n}{\alpha n}$, where $0 \leq \alpha \leq 1$.
- $\sum_{i=0}^{\alpha n} \binom{n}{i} < \binom{n}{\alpha n} \frac{1-\alpha}{1-2\alpha}$, where $0 < \alpha < 1/2$.

Proof. The first part can be verified easily using the definition of the binomial coefficients. For the second part note that

$$\sum_{i=0}^{\alpha n} \binom{n}{i} < \binom{n}{\alpha n} \cdot \left(1 + \frac{\alpha}{1-\alpha} + \left(\frac{\alpha}{1-\alpha}\right)^2 + \dots\right) = \binom{n}{\alpha n} \cdot \frac{1-\alpha}{1-2\alpha}.$$

This completes the proof. \square

As specified above, the size of each neighbor set is at least as large as the size of the set itself multiplied by some number β , which depends on the size of the set. The following lemma shows the relationship between β and the size of the set.

LEMMA 3.4. *Let S be any subset of C_n , where $|S| \leq \sum_{i=0}^{\alpha n} \binom{n}{i}$, and $0 < \alpha < 1/2$. Then $|\Gamma_S| > \beta \cdot |S|$ for $\beta = \frac{1-2\alpha}{\alpha+1/n}$.*

Proof. By Lemma 3.2 it is enough to prove the claim for sets S with size $|S| = \sum_{i=0}^{\alpha n} \binom{n}{i}$. By the isoperimetric inequality,

$$|\Gamma_S| \geq \binom{n}{\alpha n + 1}.$$

Thus we have to show that

$$\binom{n}{\alpha n + 1} > \frac{1-2\alpha}{\alpha+1/n} \cdot \sum_{i=0}^{\alpha n} \binom{n}{i}.$$

By Lemma 3.3 and the fact that $\beta = \frac{1-2\alpha}{\alpha+1/n}$ we get

$$\binom{n}{\alpha n + 1} = \binom{n}{\alpha n} \cdot \frac{1-\alpha}{1/n+\alpha} = \beta \cdot \binom{n}{\alpha n} \cdot \frac{1-\alpha}{1-2\alpha} > \beta \cdot \sum_{i=0}^{\alpha n} \binom{n}{i}$$

as claimed. \square

Proof of Theorem 1.1. The theorem is obviously true if $k = 1$ (the cube must be colored with one color) or $k = n + 1$ (each vertex is colored with a different color). Therefore assume that $1 < k < n + 1$, and let $\alpha = \frac{1}{k+1} - \frac{k-1}{n(k+1)}$. Note that by our assumption on k , $0 < \alpha < 1/2$.

Let S_i be the color sets and assume by contradiction that all sets are smaller than $\sum_{i=0}^{\alpha n} \binom{n}{i}$. We will bound $\sum_i |\Gamma_{S_i}|$ from above and below and derive a contradiction.

Consider a vertex v colored j (i.e., $v \in S_j$). The 1-neighborhood of v is k -colored, and therefore v can be a neighbor of at most $k - 1$ color sets (since it belongs to the color set S_j). Therefore each one of the 2^n vertices belongs to at most $k - 1$ of the neighborhoods Γ_{S_i} . Thus

$$\sum_i |\Gamma_{S_i}| \leq (k-1)2^n.$$

On the other hand, we can bound the number of neighbors from below as follows. If we set $\beta = k - 1$ in Lemma 3.4, then, by our choice of α , each color set S_i satisfies $|\Gamma_{S_i}| > (k-1)|S_i|$. Hence,

$$\sum_i |\Gamma_{S_i}| > \sum_i (k-1)|S_i| = (k-1)2^n,$$

which is a contradiction. Therefore there exists at least one color set with size at least $\sum_{i=0}^{\alpha n} \binom{n}{i}$, where $\alpha n = \frac{n+2}{k+1} - 1$.

3.2. Proof of Theorem 1.2. The same proof method will provide a lower bound for the more general (c, k) -coloring problem. We use the isoperimetric inequality for larger neighborhoods. Let Γ_S^c denote the c -neighborhood of a subset $S \subseteq C_n$, not including the vertices in S . That is,

$$\Gamma_S^c = \{v \notin S \mid \exists u \in S, d(u, v) \leq c\}.$$

LEMMA 3.5 (isoperimetric inequality). *Let S be a subset of C_n of size $|S| = \sum_{i=0}^{\alpha n} \binom{n}{i}$. Then $|\Gamma_S^c| \geq \sum_{i=1}^c \binom{n}{\alpha n + i}$.*

Proof. See [B, pp. 122–129]. \square

The following lemma is central to the proof of Theorem 1.2.

LEMMA 3.6. *Let S be any subset of C_n , where $|S| \leq \sum_{i=0}^{\alpha n} \binom{n}{i}$, and $0 < \alpha < 1/2$. Then $|\Gamma_S^c| > \beta \cdot |S|$ for $\beta = \left(\frac{1-2\alpha}{\alpha+c/n}\right)^c$.*

Proof. Again let S be a set of size $|S| = \sum_{i=0}^{\alpha n} \binom{n}{i}$. (The proof that for any smaller set S , the ratio $\frac{|\Gamma_S^c|}{|S|}$ can only grow is similar to that of Lemma 3.2.) Using the isoperimetric inequality we have to show that

$$\sum_{i=1}^c \binom{n}{\alpha n + i} > \left(\frac{1-2\alpha}{\alpha+c/n}\right)^c \cdot \sum_{i=0}^{\alpha n} \binom{n}{i}.$$

By applying Lemma 3.3 it is enough to show that

$$\sum_{i=1}^c \binom{n}{\alpha n + i} \geq \left(\frac{1-2\alpha}{\alpha+c/n}\right)^c \cdot \frac{1-\alpha}{1-2\alpha} \cdot \binom{n}{\alpha n}.$$

Let $a = \alpha n$. Thus, we have to prove that

$$\sum_{i=1}^c \binom{n}{a+i} \geq \left(\frac{n-2a}{a+c}\right)^c \cdot \frac{n-a}{n-2a} \cdot \binom{n}{a}.$$

Assume first that $n \geq 2a + c$. In this case we will prove the stronger inequality

$$\binom{n}{a+c} \geq \left(\frac{n-2a}{a+c}\right)^c \cdot \frac{n-a}{n-2a} \cdot \binom{n}{a}.$$

Notice that

$$\binom{n}{a+c} = \frac{n-a}{a+c} \cdot \frac{n-a-1}{a+c-1} \cdots \frac{n-a-(c-1)}{a+1} \cdot \binom{n}{a}.$$

Since $n \geq 2a + c$, then for every $0 \leq j \leq c-1$

$$\frac{n-a-j}{a+c-j} \geq \frac{n-a}{a+c}.$$

Therefore

$$\binom{n}{a+c} \geq \left(\frac{n-a}{a+c}\right)^c \cdot \binom{n}{a}.$$

We thus have to show that

$$\left(\frac{n-a}{a+c}\right)^c \geq \left(\frac{n-2a}{a+c}\right)^c \cdot \frac{n-a}{n-2a}.$$

This is obviously true for $a < n/2$, and therefore the claim follows for $n \geq 2a + c$.

Assume now that $n < 2a + c$ and recall that $a < n/2$. Thus $\binom{n}{a} \leq \binom{n}{a+1}$, and so

$$\sum_{i=1}^c \binom{n}{a+i} \geq \binom{n}{a}.$$

Since $n < 2a + c$, then $n - 2a < c$ and $n - a < a + c$. Thus,

$$\left(\frac{n-2a}{a+c}\right)^c \cdot \frac{n-a}{n-2a} \cdot \binom{n}{a} \leq \left(\frac{c}{a+c}\right)^{c-1} \cdot \binom{n}{a} \leq \binom{n}{a}.$$

This completes the proof in this case. \square

Proof of Theorem 1.2. Let S_i be the color sets, and assume by contradiction that all sets are smaller than $\sum_{i=0}^{\alpha n} \binom{n}{i}$, for $\alpha = \frac{1}{\sqrt[k-1]{2}} - \frac{c}{n} \cdot \frac{\sqrt[k-1]{c}}{\sqrt[k-1]{2}}$. Again, we can assume that $0 < \alpha < 1/2$, for otherwise the theorem is trivial.

As in Theorem 1.1, the c -neighborhood of any vertex v is k -colored, and therefore v is a neighbor of at most $k - 1$ color sets. Thus

$$\sum_i |\Gamma_{S_i}^c| \leq (k-1)2^n.$$

However, by Lemma 3.6 and the choice of α , each color set S_i satisfies $|\Gamma_{S_i}| > (k-1)|S_i|$ (simply set $\beta = k-1$ in Lemma 3.6). Again a contradiction is derived, and therefore the theorem is proved.

4. Lower bounds on the size of the largest color set in any (c, k) -coloring of $B_{n,r}$. Denote by L_r the r th layer of C_n , that is, the set of all vectors in C_n with exactly r coordinates which are 1. Let $B_{n,r} = \cup_{i=0}^r L_r$ be the Hamming ball of radius r in C_n . We now prove Theorem 1.3. A similar result can be proved for the general (c, k) -coloring. The details are omitted.

When proving Theorem 1.3, we will prove in fact a stronger claim: there exists a large color set in the r th layer of $B_{n,r}$. The proof is similar to that of Theorem 1.1; however, it uses the concept of *shadows* instead of the isoperimetric inequality. The *lower shadow* of a set $S \subseteq L_r$ is defined as

$$\partial(S) = \{v \in L_{r-1} \mid \exists u \in S, d(u, v) = 1\}.$$

LEMMA 4.1 (Kruskal–Katona). *Let $\emptyset \neq S \subseteq L_r$ be a set of size $|S| = \binom{x}{r}$ for some real number $x \geq r$. Then $|\partial(S)| \geq \binom{x}{r-1}$.*

Proof. See [B, pp. 23–39]. \square

COROLLARY 4.2. *Let $S \subseteq L_r$ be any set of size $|S| < \binom{x}{r}$ for some real number $x \geq r$. Then $|\partial(S)| > |S| \cdot r / (x - r + 1)$.*

Proof. Assume that the size of S is $|S| = \binom{y}{r}$ for $y < x$. Then, by Lemma 4.1,

$$|\partial(S)| \geq \binom{y}{r-1} = |S| \cdot \frac{r}{y-r+1} > |S| \cdot \frac{r}{x-r+1},$$

where the equality follows from the definition of the binomial coefficients, and the last inequality is true since $y < x$. \square

Proof of Theorem 1.3. Consider any $(1, k)$ -coloring of $B_{n,r}$, and let S_i be the color sets in the r th layer of $B_{n,r}$. Assume by contradiction that all sets are smaller than $\binom{n/k}{r}$. We will estimate $\sum_i |\partial(S_i)|$ and derive a contradiction.

Consider a vertex $v \in L_{r-1}$. The 1-neighborhood of v is k -colored, and therefore v can be a neighbor of at most k color sets in the r th layer. Thus v can belong to at most k shadows of sets S_i . The number of vertices in the $(r - 1)$ layer is $\binom{n}{r-1}$ and thus

$$\sum_i |\partial(S_i)| \leq k \cdot \binom{n}{r-1}.$$

On the other hand, by Corollary 4.2, $|\partial(S_i)| > |S_i| \cdot r/(n/k - r + 1)$. Also $\sum_i |S_i| = \binom{n}{r}$. Hence

$$\sum_i |\partial(S_i)| > \sum_i |S_i| \cdot \frac{r}{n/k - r + 1} = \binom{n}{r} \cdot \frac{r}{n/k - r + 1}.$$

This is a contradiction for $r \geq 1$.

5. Upper bounds for the $(1, k)$ -coloring problem. How tight is the lower bound given in Theorem 1.1? Although we do not have a full answer to this question, we show colorings of the cube for certain values of k for which the bound is tight or almost tight. We start with a few simple cases for specific values of k . In the next subsection we develop a general strategy to color the cube, which is based on *error correcting codes*.

$k = 1, n + 1$. The two extreme cases, $k = 1$ (where the cube is colored with one color) and $k = n + 1$ (where each vertex is colored with a different color), are obviously completely solved. Let us turn to more interesting colorings.

$k = 2$. In this case, we color layer j with color $\lfloor j/2 \rfloor$, $j \geq 0$. It is easy to verify that the 1-neighborhood of each vertex is colored with at most $k = 2$ colors. What is the largest color set? The color that appears the most times is the color of layers $\lfloor n/2 \rfloor$ and $\lfloor n/2 \rfloor - 1$. Thus for $k = 2$ there is a coloring of the cube in which each color appears at most $\binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor - 1}$ times. The lower bound proved in Theorem 1.1 for $k = 2$ is $\sum_{i=0}^{\lfloor (n-1)/3 \rfloor} \binom{n}{i}$. There is a gap between the upper and lower bounds, and we conjecture that the upper bound is the best possible in this case.

$k = n$. Here we can show a coloring in which each color appears at most twice. This is of course tight, since every 1-neighborhood should contain at most $k = n$ colors, and the size of a 1-neighborhood is $n + 1$. Thus there should be at least one color that appears twice. The coloring that achieves this bound is as follows.

Color every two vectors that agree on the first $n - 1$ coordinates and differ only in the last coordinate with the same color. This coloring uses 2^{n-1} different colors, and every color appears exactly twice. It remains to show that each 1-neighborhood is n -colored. However, this is clear since the neighbor of a vector x , which differs from it only in the last coordinate, is colored the same as x .

$k = n - d + 1$, where d is a constant. We can color the cube in a similar way such that each color appears 2^d times. Simply color every set of vectors that agree on all $n - d$ first coordinates with the same color. Each vector has exactly d neighbors colored as itself. Thus every 1-neighborhood is $(n - d + 1)$ -colored.

This method of coloring is efficient as long as d is constant. However, when $k \leq n/2$, a different coloring is needed in order to try and match the lower bound. For this we use error correcting codes.

5.1. Error correcting codes. Let \mathcal{C} be a binary error correcting code of length n , with minimum distance $2t + 1$ between code words. Denote by $N_t(u)$ the sphere

of radius t around a code word u , i.e., all vectors of distance at most t from u . Such a code is called an (n, t) -code (see [PW] or [MS] for a comprehensive description of error correcting codes). Given any (n, t) -code we define the *induced coloring* of C_n as follows:

Color each code word with a different color. Vectors not in the code are colored with the color of the nearest code word, breaking ties arbitrarily.

Notice that each sphere $N_t(u)$ around a code word u is colored with the color of the code word u . Thus each color appears at least $\sum_{i=0}^t \binom{n}{i}$ times, and the 1-neighborhoods of vectors at distance less than t from some code word are 1-colored. The question is what happens with vectors at distance greater than t from any code word. We first look at some special codes that provide better upper bounds.

5.1.1. Perfect codes. An (n, t) -perfect code is a code for which the spheres of radius t around the code words partition C_n . Thus every vector is at distance at most t from some code word. Perfect codes exist only for very restricted values of n and t . However, since the same proof method will be used subsequently to color the cube using a general code, assume for the moment that we have a perfect code for any n and t .

Begin with the induced coloring of some (n, t) -perfect code. Since this is a perfect code, each color appears exactly $\sum_{i=0}^t \binom{n}{i}$ times. It remains to bound the number of colors in the 1-neighborhood of each vector.

The 1-neighborhood of any vector x at distance less than t from some code word is 1-colored. Therefore consider only vectors x at distance exactly t from some code word u (i.e., x is on the boundary of the sphere $N_t(u)$). The vector x is colored the same as u , and it has exactly t neighbors in $N_t(u)$ that are colored with the same color as it is.

In how many colors are the remaining $n - t$ neighbors of x (those not in $N_t(u)$) colored? We claim that they are colored in at most $(n - t)/(t + 1)$ colors. To see this, note that each one of these neighbors belongs to some sphere $N_t(v)$ around a code word v , where v is at distance $t + 1$ from x . Furthermore, there are exactly $t + 1$ neighbors of x in any such sphere. Thus the neighbors of x that are not in $N_t(u)$ can be divided into equivalence classes of size $t + 1$, according to the spheres around code words to which they belong.

Therefore the $n - t$ neighbors of x that are not in $N_t(u)$ are colored in $(n - t)/(t + 1)$ colors. If we add to this the color of x and the t neighbors that are in $N_t(u)$, we get a total of $k = (n - t)/(t + 1) + 1 = (n + 1)/(t + 1)$ colors in the 1-neighborhood of x . We have thus proved the following:

Any (n, t) -perfect code induces a $(1, k)$ -coloring of C_n , in which each color appears $\sum_{i=0}^t \binom{n}{i}$ times, where $t = \frac{n+1}{k} - 1$.

As we can see, this result almost matches the lower bound stated in Theorem 1.1. Unfortunately, perfect error correcting codes are rather rare. The Hamming code is perfect for $t = 1$ (i.e., $k = \frac{n+1}{2}$), and n of the form $n = 2^r - 1$, for some r . The Golay code is perfect for $n = 23$ and $t = 3$.

COROLLARY 5.1.

1. *For any $n = 2^r - 1$, there exists a $(1, \frac{n+1}{2})$ -coloring of C_n in which each color appears $n + 1$ times.*
2. *There exists a $(1, 6)$ -coloring of C_{23} in which each color appears 24 times.*

However, for larger values of t the situation is much worse. In fact it was proven that no other nontrivial perfect codes exist (see [V]). The resort is to examine larger classes of codes.

5.1.2. Quasi-perfect codes. An (n, t) -quasi-perfect code is a code for which the spheres of radius t around code words are disjoint, and every vector is at distance at most $t + 1$ from some code word. A subclass of quasi-perfect codes are *nearly perfect codes*. These codes were defined by Goethals and Snover, and their exact definition can be found in [GS].

Goethals and Snover [GS] list a few nearly perfect codes. For $t = 1$ there exists a nearly perfect code for any $n = 2^r - 2$. For $t = 2$ there exists a nearly perfect code for any $n = 4^r - 1$. Lindstrom proved that no other nearly perfect codes exist [L]. The following lemma is from [GS].

LEMMA 5.2 (see [GS]). *For any (n, t) -nearly perfect code,*

1. *any vector at distance greater than t from any code word is at distance $t + 1$ from exactly $\lfloor n/(t + 1) \rfloor$ code words;*
2. *any vector at distance t from some code word is at distance $t + 1$ from exactly $\lfloor (n - t)/(t + 1) \rfloor$ other code words.*

COROLLARY 5.3. *For any $n = 4^r - 1$, there exists a $(1, 1 + \frac{n}{3})$ -coloring of C_n in which each color appears $O(n^3)$ times.*

Proof. Take an (n, t) -nearly perfect code, where $t = 2$ and $n = 4^r - 1$, and consider the induced coloring. Since this is a nearly perfect code, each word is at distance at most $t + 1$ from some code word. Thus each color appears at most $\sum_{i=0}^{t+1} \binom{n}{i} = O(n^3)$ times.

We now have to show how many colors appear in each 1-neighborhood. The interesting cases are those of vectors at distance t and $t + 1$ from some code word. Let x be a vector at distance $t + 1$ from some code word. Notice that each code word u that is at distance $t + 1$ from x forces $t + 1$ of the neighbors of x to be colored in the same color as u . By Lemma 5.2 there are $\lfloor n/(t + 1) \rfloor$ code words at distance $t + 1$ from x . Since $n = 4^r - 1$ and $t + 1 = 3$, it is easy to verify that $t + 1$ divides n . Therefore the neighborhood of x is colored with $k = 1 + n/(t + 1)$ colors. (We add the color of x .) The proof for vectors at distance t from some code word is similar. \square

Similar results can be obtained for any quasi-perfect code. It is not known yet whether quasi-perfect codes exist for large values of t . Therefore we state the general theorem for any (n, t) -quasi-perfect code with the hope that more codes will be found.

As before, take any (n, t) -quasi-perfect code and look at the induced coloring. Each vector is at distance at most $t + 1$ from some code word, and so each color appears at most $\sum_{i=0}^{t+1} \binom{n}{i}$ times. We have only to bound the number of colors in each 1-neighborhood. Again, the interesting cases are those of vectors at distance t or $t + 1$ from some code word.

LEMMA 5.4. *The 1-neighborhood of each vector is colored with at most $k = O(n^2/t^2)$ colors.*

Proof. Let x be some vector. Denote by Z_i the number of code words at distance $t + i$ from x , where $i = 1, 2$. The number of colors in the 1-neighborhood of x does not exceed $1 + Z_1 + Z_2$, since the neighbors of x receive their colors from code words at distance $t + 1$ and $t + 2$ from x , and we have to add the color of x itself.

There are $\binom{n}{i}$ vectors at distance i from x . Call them the i -neighbors of x . Let v be some code word at distance $t + i$ from x . Then, exactly $\binom{t+i}{i}$ of the i -neighbors of x belong to $N_t(v)$. Since the spheres of radius t around code words are disjoint, there are at most $\binom{n}{i} / \binom{t+i}{i}$ code words at distance $t + i$ from x . Thus the number of colors in the 1-neighborhood of x is at most $k \leq 1 + \binom{n}{1} / \binom{t+1}{1} + \binom{n}{2} / \binom{t+2}{2} = O(n^2/t^2)$ as stated. \square

THEOREM 5.5. *Any (n, t) -quasi-perfect code induces a $(1, k)$ -coloring of C_n , in*

which each color appears at most $\sum_{i=0}^{t+1} \binom{n}{i}$ times, where $t = O(n/\sqrt{k})$.

5.1.3. General codes. The method of proof used for quasi-perfect codes can be generalized for any code. An (n, t, d) -code is a code in which the spheres of radius t around code words are disjoint, and the spheres of radius $t + d$ around code words cover the whole cube. Thus for perfect codes $d = 0$, and for quasi-perfect codes, $d = 1$.

It is again possible to look at the induced coloring of such a code and bound the number of colors in each 1-neighborhood by $k = O((n/t)^{d+1})$. Then a similar theorem can be proved.

THEOREM 5.6. *Any (n, t, d) -code induces a $(1, k)$ -coloring of C_n , in which each color appears at most $\sum_{i=0}^{t+d} \binom{n}{i}$ times, where $t = O(n/k^{1/(d+1)})$.*

Similar techniques can be used to show upper bounds for the (c, k) -coloring problem. The details are omitted.

6. Generalizations and further research. We have proved lower and upper bounds on the (c, k) -coloring problem, in which each word was hashed exactly once to the hash table. It may be useful to allow each word to be hashed to s entries in the hash table, using s hash functions. More formally, a (c, k, s) -covering of Σ^n is a collection of subsets S_i that cover Σ^n such that

- each word $u \in \Sigma^n$ is contained in at most s of the subsets S_i ;
- for any $u \in \Sigma^n$, there exist k subsets S_{u_1}, \dots, S_{u_k} such that $N_c(u) \subseteq \cup_{i=1}^k S_{u_i}$.

The (c, k, s) -covering problem is to minimize the size of the largest subset S_i .

The solution we showed for the (c, k) -coloring problem may give insight into the solution of the general (c, k, s) -covering problem. The lower bound for the (c, k) -coloring problem was proved using the isoperimetric inequality. This inequality is tight for sets that are spheres and only for them (see [B]). On the other hand, the upper bound shows that a perfect code that covers the cube with disjoint spheres induces a $(1, k)$ -coloring that almost matches the lower bound. These observations make it plausible that in an optimal $(1, k)$ -coloring, all color sets have the structure of spheres. We also showed a tradeoff between k and the size of the color sets (or the radius t of the spheres) (see Theorems 1.1 and 1.2).

All this suggests that in order to find a (c, k, s) -covering, it may be wise to try and cover Σ^n with spheres (possibly overlapping). By changing the radius of the spheres, it may be possible to find a general tradeoff between s , k , and the size of the largest subset.

For example, suppose we want to find a $(1, 1, s)$ -covering of C_n . It is possible to simply cover the whole cube with one subset. In this case $s = 1$, but there is one large subset. Or we can cover the cube by defining a different subset for the 1-neighborhood of each vector. In this case the size of each subset is optimal, but s is large. Instead, we can try to combine these two methods as follows.

Let \mathcal{C} be an (n, t) -perfect code. Define a subset $S_u = N_{t+1}(u)$ for each $u \in \mathcal{C}$. The coverings described above are extreme cases of this covering, with $t = n$ and $t = 0$. The size of each subset is $\sum_{i=0}^{t+1} \binom{n}{i}$, and it is possible to show that $s = (n+1)/(t+1)$. Thus by choosing t , it is possible to get the desired tradeoff between the size of each subset and s . (A similar method was used by [DHP] to design algorithms for the retrieval of neighbors from dictionaries. See also [H] and [P].) This method can be generalized to cover the cube using a general (n, t, d) -code, where the efficiency of the cover depends on d .

Several interesting open questions remain. The extensions of the upper bounds for the (c, k) -coloring problem and the (c, k, s) -covering problems use a general (n, t, d) -code, and their efficiency depends on d . Thus an important open question is to try

and determine an upper bound on d that would guarantee the existence of an (n, t, d) -code for any given n and t . An upper bound of $d = t$ is easy to show, but no better asymptotically general bound is known. This question is of independent interest to the study of error correcting codes, and some bounds were given for special types of codes (see [CKMS], [VS]).

There are also open problems concerning the $(1, k)$ -coloring problem. The upper bound for this problem can probably be improved when k is a constant. The lower bound seems to be tight or almost tight for all cases, except for $k = 2$. We conjecture that for $k = 2$, the upper bound we showed of $O(\binom{n}{n/2})$ is the best possible.

Finally, it would be interesting to check what happens with different distance measures (not necessarily the Hamming distance). Different distance measures induce other graphs (other than the n -dimensional cube) for which one can try to solve the coloring or covering problems. Expanders may be an interesting class of graphs to check.

Acknowledgment. We would like to thank the anonymous referees for their comments, which improved the presentation of this paper.

REFERENCES

- [B] B. BOLLOBAS, *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability*, Cambridge University Press, Cambridge, UK, 1986.
- [CKMS] G.D. COHEN, M.G. KARPOVSKY, H.F. MATTSON, AND J.R. SCHATZ, *Covering radius—survey and recent results*, IEEE Trans. Inform. Theory, 31 (1985), pp. 328–343.
- [DHP] D. DOLEV, Y. HARARI, AND M. PARNAS, *Finding the neighborhood of a query in a dictionary*, in Proceedings of the 2nd Israel Symposium on Theory of Computing and Systems, 1993, pp. 33–42.
- [GS] J.M. GOETHALS AND S.L. SNOVER, *Nearly perfect binary codes*, Discrete Math., 3 (1972), pp. 65–88.
- [H] Y. HARARI, *Algorithms and Lower Bounds for the Retrieval of Neighbors from a Dictionary*, M.Sc. thesis, Hebrew University, Jerusalem, Israel, 1992 (in Hebrew).
- [L] K. LINDSTROM, *The nonexistence of unknown nearly perfect codes*, Ann. Univ. Turku., Ser. A I, 169 (1975), pp. 3–28.
- [LV1] G.M. LANDAU AND U. VISHKIN, *Efficient string matching in the presence of errors*, in Proceedings of the 26th IEEE Symposium on Foundations of Computer Science, 1985, pp. 126–136.
- [LV2] G.M. LANDAU AND U. VISHKIN, *Introducing efficient parallelism into approximate string matching and a new serial algorithm*, in Proceedings of the 18th Annual ACM Symposium on Theory of Computing, 1986, pp. 220–230.
- [MS] F.J. MACWILLIAMS AND N.J.A. SLOANE, *The Theory of Error Correcting Codes*, North-Holland, Amsterdam, 1977.
- [P] M. PARNAS, *Robust Algorithms and Data Structures for Information Retrieval*, Ph.D. thesis, Hebrew University, Jerusalem, Israel, 1994.
- [PW] W.W. PETERSON AND E.J. WELDON, *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972.
- [R] R.L. RIVEST, *On hash-coding algorithms for partial-match retrieval*, in Proceedings of the 15th Annual Symposium on Switching and Automata Theory, IEEE, 1974, pp. 95–103.
- [SK] D. SANKOFF AND J.B. KRUSKAL, *Time Warps, Strings Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA, 1983.
- [V] J.H. VAN LINT, *A survey of perfect codes*, Rocky Mountain J. Math., 3 (1975), pp. 199–224.
- [VS] S.G. VLADUTS AND A.N. SKOROBOGATOV, *Covering radius for long BCH codes*, Problemy Peredachi Informatsii, 25 (1989), pp. 38–45.

ON COVERING A BIPARTITE GRAPH WITH CYCLES*

HONG WANG[†]

Abstract. We conjectured in [H. Wang, *Australas. J. Combin.*, 19 (1999), pp. 115–121] that, for each integer $k \geq 2$, there exists $N(k)$ such that if $G = (V_1, V_2; E)$ is a bipartite graph with $|V_1| = |V_2| = n \geq N(k)$ and $d(x) + d(y) \geq n + k$ for each pair of nonadjacent vertices x and y of G with $x \in V_1$ and $y \in V_2$, then for any k independent edges e_1, \dots, e_k of G , there exist k vertex-disjoint cycles C_1, \dots, C_k in G such that $e_i \in E(C_i)$ for all $i \in \{1, \dots, k\}$ and $V(C_1 \cup \dots \cup C_k) = V(G)$. This conjecture is also verified for $k = 2$ in [H. Wang, *Australas. J. Combin.*, 19 (1999), pp. 115–121]. We prove this conjecture for $k = 3$ in this paper.

Key words. bipartite graphs, cycles, coverings

AMS subject classifications. 05C35, 05C70

PII. S0895480196310487

1. Introduction. We discuss only finite simple graphs and use standard terminology and notation from [2] except as indicated. The following is conjectured in [5].

CONJECTURE A (see [5]). *For each integer $k \geq 2$, there exists $N(k)$ such that if $G = (V_1, V_2; E)$ is a bipartite graph with $|V_1| = |V_2| = n \geq N(k)$ and $d(x) + d(y) \geq n + k$ for each pair of nonadjacent vertices x and y of G with $x \in V_1$ and $y \in V_2$, then for any k independent edges e_1, \dots, e_k of G , there exist k vertex-disjoint cycles C_1, \dots, C_k in G such that $e_i \in E(C_i)$ for all $i \in \{1, \dots, k\}$ and $V(C_1 \cup \dots \cup C_k) = V(G)$.*

As pointed out in [5], if this conjecture is true, the condition on the degrees of G is sharp. To see this, let $G = (X, Y; E)$ be a bipartite graph obtained from a complete bipartite graph $K_{n-1, n}$ by adding a new vertex x_0 to $K_{n-1, n}$ such that $N_G(x_0) = \{x_1, x_2, \dots, x_k\}$, where x_1, x_2, \dots, x_k are k distinct vertices of $K_{n-1, n}$ whose degrees in $K_{n-1, n}$ are $n - 1$. Then for each pair of nonadjacent vertices x and y of G with $x \in X$ and $y \in Y$, we have $x_0 \in \{x, y\}$ and $d(x) + d(y) = n + k - 1$. Let e_1, \dots, e_k be k independent edges in G such that e_i is incident with x_i for all $i \in \{1, \dots, k\}$ and $e_1 = x_0x_1$. Clearly, every cycle passing through e_1 must contain at least three vertices in $\{x_0, x_1, \dots, x_k\}$. Therefore there are no k vertex-disjoint cycles in G satisfying the requirement.

This conjecture is verified for $k = 2$ in [5]. To state the result, let F be a graph obtained from $K_{4,4}$ by removing three independent edges from $K_{4,4}$.

THEOREM B (see [5]). *Let $G = (V_1, V_2; E)$ be a bipartite graph with $|V_1| = |V_2| = n \geq 4$. Suppose $d(x) + d(y) \geq n + 2$ for each pair of nonadjacent vertices x and y of G with $x \in V_1$ and $y \in V_2$. Then for any two independent edges e_1 and e_2 of G , G has two vertex-disjoint cycles C_1 and C_2 such that $e_i \in E(C_i)$ for each $i \in \{1, 2\}$ and $V(C_1 \cup C_2) = V(G)$, unless G is isomorphic to F .*

In this paper, we prove the conjecture for $k = 3$. To state the result, we define the three following bipartite graphs. First, for any two graphs H and G , if we write $H \leq G$, it means that H is isomorphic to a subgraph of G . Let (X, Y) be the bipartition of $K_{6,6}$. Let F_0 be a spanning subgraph of $K_{6,6}$ such that F_0 consists of three paths of length 2 and three isolated vertices from X . Let F_1 be a spanning

*Received by the editors October 11, 1996; accepted for publication (in revised form) October 24, 2001; published electronically January 4, 2002.

<http://www.siam.org/journals/sidma/15-1/31048.html>

[†]Department of Mathematics, University of Idaho, Moscow, ID 83844 (hwang@uidaho.edu).

subgraph of $K_{6,6}$ such that F_1 consists of four paths of length 2. Let F_2 be a spanning subgraph of $K_{6,6}$ such that F_2 consists of two paths of length 2, a path of length 1, two isolated vertices from X , and two isolated vertices from Y . Let F_3 be a spanning subgraph of $K_{7,7}$ such that F_3 consists of three cycles of length 4 and two isolated vertices. Let $G_0 = K_{6,6} - E(F_0)$, $G_1 = K_{6,6} - E(F_1)$, $G_2 = K_{6,6} - E(F_2)$, and $G_3 = K_{7,7} - E(F_3)$. It is not difficult to check that for each $i \in \{1, 2, 3\}$, there exist three independent edges e_1, e_2 , and e_3 in G_i such that G_i does not contain three vertex-disjoint cycles C_1, C_2 , and C_3 with $e_j \in E(C_j)$ for $j \in \{1, 2, 3\}$ and $V(C_1 \cup C_2 \cup C_3) = V(G_i)$. For instance, let $X = \{x_1, \dots, x_6\}$, $Y = \{y_1, \dots, y_6\}$, and $E(F_2) = \{x_1y_2, x_1y_3, y_1x_2, y_1x_3, x_4y_4\}$. Choose $e_1 = x_1y_1$, $e_2 = x_5y_5$, and $e_3 = x_6y_6$. Then G_2 does not contain an e_1 -cycle C of length 4 such that $V(C) \cap V(e_2 \cup e_3) = \emptyset$.

We prove the following result.

THEOREM C. *Let $G = (V_1, V_2; E)$ be a graph with $|V_1| = |V_2| = n \geq 6$. Suppose $d(x) + d(y) \geq n + 3$ for each pair of nonadjacent vertices x and y of G with $x \in V_1$ and $y \in V_2$. Then for any three independent edges e_1, e_2 , and e_3 , G has three vertex-disjoint cycles C_1, C_2 , and C_3 such that $e_i \in E(C_i)$ for each $i \in \{1, 2, 3\}$ and $V(C_1 \cup C_2 \cup C_3) = V(G)$ unless either $G \cong G_0$, or $G_1 \leq G \leq G_2$, or $G \cong G_3$.*

We shall use the following terminology and notation. Let G be a graph. For a vertex $u \in V(G)$ and a subgraph H of G , $N(u, H)$ is the set of neighbors of u contained in H , i.e., $N(u, H) = N(u) \cap V(H)$. We let $d(u, H) = |N(u, H)|$. Thus $d(u, G)$ is the degree of u in G . For a subset U of $V(G)$, $G[U]$ denotes the subgraph of G induced by U . Let e be an edge of G . An e -subgraph of G is a subgraph H of G such that $e \in E(H)$. In particular, an e -Hamiltonian cycle or path of G is a Hamiltonian cycle or path of G passing through e , respectively. If P is an e -path, we define $\sigma(e, P) = \min(|E(P')|, |E(P'')|)$, where P' and P'' are two components of $P - e$. If P is of odd length, say $P = x_1x_2 \dots x_{2q}$, we define $E_0(P) = \{x_1x_2, x_{2q-1}x_{2q}\} \cup \{x_i x_{i+1} | i = 2, 4, \dots, 2q-2\}$ and $E_1(P) = \{x_j x_{j+1} | j = 3, 5, \dots, 2q-3\}$, and, moreover, let $r(e, P) = 0$ if $e \in E_0(P)$ and $r(e, P) = 1$ if $e \in E_1(P)$. We use $l(C)$ and $l(P)$ to denote the length of a cycle C and the length of a path P , respectively. An edge is also considered as a path of length 1. We define $\delta_2(G)$ to be the minimum of $d(x) + d(y)$ for all nonadjacent pairs of vertices x and y in G with $x \in V_1$ and $y \in V_2$. As usual, $\delta(G)$ is the minimum degree of G .

If $G = (V_1, V_2; E)$ is a connected bipartite graph, we define $\overline{G}^b = (V_1, V_2; E')$, where $E' = \{xy | xy \notin E, x \in V_1, \text{ and } y \in V_2\}$.

2. Lemmas. The following lemmas are Ore-type lemmas in bipartite graphs. Their proofs can be found in or easily deduced from [1, 3, 4]. For the sake of completeness, we include their proofs. Let $G = (V_1, V_2; E)$ be a given bipartite graph in the following.

LEMMA 2.1. *Let e be an edge and $P = x_1x_2 \dots x_{2q}$ an e -path in G . Let $y \in V(G) - V(P)$ such that $\{x_{2q}, y\} \not\subseteq V_i$ for every $i \in \{1, 2\}$. If $d(x_{2q}, P) + d(y, P) \geq q + 1 + r(e, P)$, then G has an e -path P' such that $V(P') = V(P) \cup \{y\}$. Moreover, if $e \neq x_1x_2$, then P' is a path from y to x_1 .*

Proof. Clearly, the lemma holds if $yx_{2q} \in E$. So we may assume $yx_{2q} \notin E$. As $d(y, P) > 0$, it is also easy to see that if $e = x_1x_2$ and $x_1x_{2q} \in E$, then the lemma holds. Hence we may assume that if $e = x_1x_2$, then $x_1x_{2q} \notin E$. Let $I = \{x_{i+1} | x_i x_{2q} \in E\}$. Then $|N(y, G) \cap I| = |N(y, G)| + |I| - |N_G(y) \cup I| \geq q + 1 + r(e, P) - q = 1 + r(e, P)$. This implies that there exists $\{x_{i+1}, x_{j+1}\} \subseteq N_G(y) \cap I$ with $i \neq j$ if $r(e, P) = 1$. By our assumption, we see that if $r(e, P) = 0$, then $e \notin \{x_i x_{i+1}, x_j x_{j+1}\}$. We may assume without loss of generality (w.l.o.g.) that $x_i x_{i+1} \neq e$ if $i \neq j$. Then $P' =$

$yx_{i+1}x_{i+2}\dots x_{2q}x_ix_{i-1}\dots x_1$ is a required path. \square

LEMMA 2.2. *Let e be an edge and $P = x_1x_2\dots x_{2q}$ an e -path with $q \geq 2$ in G . If $d(x_1, P) + d(x_{2q}, P) \geq q + 1 + r(e, P)$, then G has an e -cycle C with $V(C) = V(P)$.*

Proof. Clearly, the lemma holds if $x_1x_{2q} \in E$. So we may assume $x_1x_{2q} \notin E$. The condition implies that there exist x_i and x_j for some $\{i, j\} \subseteq \{1, 3, \dots, 2q-1\}$ such that $\{x_1x_{i+1}, x_{2q}x_i, x_1x_{j+1}, x_{2q}x_j\} \subseteq E$ with $i \neq j$ if $r(e, P) = 1$. As $x_1x_{2q} \notin E$, we see that $e \notin \{x_ix_{i+1}, x_jx_{j+1}\}$ if $r(e, P) = 0$. We may assume w.l.o.g. that $e \neq x_ix_{i+1}$ if $i \neq j$. Then $C' = x_1x_2\dots x_ix_{2q}x_{2q-1}\dots x_{i+1}x_1$ is a required cycle. \square

Remark. From the proof of Lemma 2.2, we see that if $d(x_1, P) + d(x_{2q}, P) \geq q + 1$ and G does not have an e -cycle C with $V(C) = V(P)$, then $d(x_1, P) + d(x_{2q}, P) = q + 1$, $r(e, P) = 1$, and $\{x_1x_{2a+2}, x_{2q}x_{2a+1}\} \subseteq E$, where $e = x_{2a+1}x_{2a+2}$.

LEMMA 2.3. *Let e be an edge and C an e -cycle in G . Let $y \in V(G) - V(C)$. If $d(y, C) \geq 2$, then $G[V(C) \cup \{y\}]$ contains an e -cycle C' such that $l(C') < l(C)$, unless $d(y, C) = 2$, and, moreover, if $N(y, C) = \{x', x''\}$, then C has a subpath $x'zx''$ with z not incident with e .*

Proof. Say $C = x_1x_2\dots x_{2q}x_1$ with $e = x_1x_{2q}$. Let $\{x_i, x_j\} \subseteq N(y, C)$ such that $1 \leq i < j \leq 2q$ and $xy \notin E$ for all $x \in V(C) - \{x_i, x_{i+1}, \dots, x_j\}$. Clearly, $C' = x_1\dots x_ix_j\dots x_{2q}x_1$ is an e -cycle. If $l(C') \not< l(C)$, then $j = i + 2$ and the lemma follows. \square

LEMMA 2.4. *Let e be an edge, C an e -cycle, and P a path with two endvertices $u \in V_1$ and $v \in V_2$ in G such that $V(C) \cap V(P) = \emptyset$. Let $l(C) = 2q$. If $d(u, C) + d(v, C) \geq q + 1$, then G has an e -cycle C' with $V(C') = V(C \cup P)$.*

Proof. Let $C = x_1x_2\dots x_{2q}x_1$ with $e = x_1x_{2q}$ and $x_1 \in V_1$. The condition implies that $\{x_iv, x_{i+1}u\} \subseteq E$ for some $i \in \{1, 3, \dots, 2q-1\}$. Then $x_1x_{2q}x_{2q-1}\dots x_{i+1}uPx_ix_{i-1}\dots x_1$ is a required cycle. \square

LEMMA 2.5. *If $|V_1| = |V_2|$ and $\delta_2(G) \geq |V_1|$, then G has a perfect matching.*

Proof. Let $\{f_1, f_2, \dots, f_m\}$ be a maximum matching in G . If $m < |V_1|$, let $x \in V_1$ and $y \in V_2$ be two vertices which are not incident with any edge in $\{f_1, f_2, \dots, f_m\}$. As $d(x) + d(y) \geq |V_1|$, we see that there exists f_i such that $d(x, f_i) + d(y, f_i) = 2$. Obviously, $G[V(f_i) \cup \{x, y\}]$ contains two independent edges, and therefore $\{f_1, f_2, \dots, f_m\}$ is not a maximum matching of G , a contradiction. \square

3. Proof of Theorem C. The proof is self-contained. Let $G = (V_1, V_2; E)$ be a bipartite graph with $|V_1| = |V_2| = n \geq 6$ and $\delta_2(G) \geq n + 3$. Suppose that there exist three independent edges e_1, e_2 , and e_3 of G such that G does not have three vertex-disjoint cycles C_1, C_2 , and C_3 with $e_i \in E(C_i)$ for each $i \in \{1, 2, 3\}$ and $V(C_1 \cup C_2 \cup C_3) = V(G)$. We shall prove that either $G_1 \leq G \leq G_2$ or $G \cong G_3$.

Note that as $\delta_2(G) \geq n + 3$, we have $\delta(G) \geq 4$. We first prove the three following claims.

Claim 1. $G - V(e_1 \cup e_2 \cup e_3)$ has a perfect matching.

Proof of Claim 1. Clearly, $\delta_2(G - V(e_1, e_2, e_3)) \geq n + 3 - 6 = n - 3$. By Lemma 2.5, $G - V(e_1 \cup e_2 \cup e_3)$ has a perfect matching.

Claim 2. For each $i \in \{1, 2, 3\}$, there exists an e_i -cycle C_i in G such that $l(C_i) \leq 6$ and $V(C_i) \cap V(e_j \cup e_k) = \emptyset$, where $\{i, j, k\} = \{1, 2, 3\}$.

Proof of Claim 2. Enumerate $V_1 = \{x_1, x_2, \dots, x_n\}$ and $V_2 = \{y_1, y_2, \dots, y_n\}$ such that $\{x_t y_t \mid 1 \leq t \leq n\}$ is a perfect matching in G with $e_t = x_t y_t$ ($t = 1, 2, 3$). As $\delta(G) \geq 4$, we may assume $x_i y_4 \in E$. If $y_i x_4 \in E$, we are done. So assume $y_i x_4 \notin E$. As $d(y_i) + d(x_4) \geq n + 3$, there exists $t \in \{5, \dots, n\}$ such that $d(y_i, x_t y_t) + d(x_4, x_t y_t) = 2$. Thus $x_i y_i x_t y_t x_4 y_4 x_i$ is a required cycle. \square

Claim 3. For some $\{i, j, k\} = \{1, 2, 3\}$, there exist two vertex-disjoint cycles C_1 and C_2 of length at most 6 such that $e_i \in E(C_1)$, $e_j \in E(C_2)$, and $V(e_k) \cap V(C_1 \cup C_2) = \emptyset$.

Proof of Claim 3. We choose a smallest cycle C_1 such that $e_1 \in E(C_1)$ and $V(C_1) \cap V(e_2 \cup e_3) = \emptyset$, and, subject to this, we further choose C_1 such that the number of edges in a maximum matching of $G - V(C_1 \cup e_2 \cup e_3)$ is maximal. The existence of C_1 is guaranteed by Claim 2. Thus $l(C_1) \leq 6$. Let $F = \{a_1b_1, \dots, a_mb_m\}$ be a maximum matching of $G - V(C_1 \cup e_2 \cup e_3)$ with $\{a_1, \dots, a_m\} \subseteq V_1$. If it does not have a perfect matching, then $G - V(C_1 \cup e_2 \cup e_3)$ has two nonadjacent vertices x and y with $x \in V_1$ and $y \in V_2$ such that neither of x and y is incident with an edge in F . By the maximality of F , $d(x, a_tb_t) + d(y, a_tb_t) \leq 1$ for all $t \in \{1, 2, \dots, m\}$. By Lemma 2.3 and the choice of C_1 , we have $d(x, C_1) \leq 2$ and $d(y, C_1) \leq 2$. Clearly, $d(x, e_2 \cup e_3) + d(y, e_2 \cup e_3) \leq 4$. As $d(x) + d(y) \geq n + 3$, it follows that $l(C_1) = 4$ and $d(x, C_1) = d(y, C_1) = 2$. Let $z \in V(C_1) \cap V_1$ be such that z is not incident with e_1 . Then $yz \in E - F$ and $C'_1 = C_1 - z + x$ is an e_1 -cycle of length 4, contradicting the choice of C_1 . Therefore F is a perfect matching of $G - V(C_1 \cup e_2 \cup e_3)$.

On the contrary, we suppose that Claim 3 is not true. Let $e_2 = uv$ and $e_3 = xy$ with $\{u, x\} \subseteq V_1$. We readily see that $d(w, e_2 \cup e_3) > 0$ for some $w \in V(G) - V(C_1 \cup e_2 \cup e_3)$ because $d(u) + d(v) \leq n + 2$ otherwise. W.l.o.g., we assume $b_1u \in E$. Clearly, $a_1v \notin E$. If there exists some $t \in \{2, 3, \dots, m\}$ such that $d(a_1, a_tb_t) + d(v, a_tb_t) = 2$, then $b_1u \in E$ by the choice of b_1 and so C_1 and $uva_tb_1a_1b_1u$ are two required cycles, a contradiction. Hence $d(a_1, e_t) + d(v, e_t) \leq 1$ for each $t \in \{2, 3, \dots, m\}$. By Lemma 2.3 and the choice of C_1 , we have $d(a_1, C_1) \leq 2$. As $d(a_1) + d(v) \geq n + 3$, it follows that $d(v, C_1) = l(C_1)/2$, $vx \in E$, $a_1y \in E$, and $d(a_1, C_1) = 2$. Then $b_1x \notin E$. Similarly, as $a_1y \in E$ we can show that $d(x, C_1) = l(C_1)/2$ and $d(b_1, C_1) = 2$. If $l(C_1) = 4$, let $C_1 = w_1w_2w_3w_4w_1$ with $w_1 \in V_1$ and $e_1 = w_1w_4$, and then xya_1w_2x and $w_1w_4w_3b_1w_1$ are two required cycles, a contradiction. Hence $l(C_1) = 6$. Let $C_1 = w_1w_2w_3w_4w_5w_6w_1$ with $w_1 \in V_1$ and $e_1 = w_1w_6$. By the choice of C_1 and Lemma 2.3, we have $a_1w_4 \in E$ and $b_1w_3 \in E$. If $b_1w_5 \in E$, then $C_1 - w_4 + b_1$ and xya_1w_4x are two required cycles, a contradiction. Hence $b_1w_1 \in E$. Similarly, $a_1w_6 \in E$. Therefore $C'_1 = w_1w_6a_1b_1w_1$ is an e_1 -cycle with $l(C'_1) < l(C_1)$, contradicting the choice of C_1 . So the claim holds. \square

By Claim 3, we choose $\{i, j, k\} = \{1, 2, 3\}$ and two cycles C_1 and C_2 such that

$$(1) \quad \begin{aligned} e_i \in E(C_1), e_j \in E(C_2), \quad V(C_1) \cap V(C_2) = \emptyset, \\ V(e_k) \cap V(C_1 \cup C_2) = \emptyset, \quad l(C_1) \leq 6, \quad \text{and } l(C_2) \leq 6. \end{aligned}$$

Subject to (1), we choose $\{i, j, k\}$, C_1 , and C_2 such that

$$(2) \quad l(C_1) + l(C_2) \text{ is minimal.}$$

Subject to (1) and (2), we further choose $\{i, j, k\}$, C_1 and C_2 such that

$$(3) \quad \text{The length of a longest path containing } e_k \text{ in } G - V(C_1 \cup C_2) \text{ is maximal.}$$

Let P be a longest e_k -path in $G - V(C_1 \cup C_2)$. Subject to (1), (2), and (3), we finally choose $\{i, j, k\}$, C_1 , C_2 , and P such that

$$(4) \quad \sigma(e_k, P) \text{ is minimal.}$$

W.l.o.g., say $i = 1$, $j = 2$, and $k = 3$. Let $C_1 = x_1x_2 \dots x_{2t_1}x_1$ and $C_2 = y_1y_2 \dots y_{2t_2}y_1$ with $e_1 = x_1x_{2t_1}$ and $e_2 = y_1y_{2t_2}$. Let $P = u_1u_2 \dots u_s$ and $H =$

$G - V(C_1 \cup C_2)$. W.l.o.g., say $\{x_1, y_1, u_1\} \subseteq V_1$. By our assumption on G , H does not have an e_3 -Hamiltonian cycle. Set $2t = |V(H)|$. Note that by (2), neither C_1 nor C_2 has a chord in G . By (2) and Lemma 2.3, we have

$$(5) \quad d(u, C_i) \leq 2 \text{ for each } i \in \{1, 2\} \text{ and } u \in V(H) - V(e_3).$$

With the choice of C_1 , C_2 , and P , we prove the following two claims.

Claim 4. $V(P) = V(H)$, i.e., $s = 2t$.

Proof of Claim 4. Suppose $s < 2t$. We distinguish two cases: s is even or s is odd.

Case a. $s = 2k$. Choose a vertex $u_0 \in V(H) - V(P)$ with $u_0 \in V_1$. By (3) and Lemma 2.1, $d(u_0, P) + d(u_{2k}, P) \leq k + r(e_3, P)$, and so $d(u_0, H) + d(u_{2k}, H) \leq t + r(e_3, P)$. As $\delta_2(G) \geq n + 3$, it follows that $d(u_0, C_1 \cup C_2) + d(u_{2k}, C_1 \cup C_2) \geq t_1 + t_2 + 3 - r(e_3, P)$. Suppose first that, for some $i \in \{1, 2\}$, say $i = 1$, $d(u_0, C_1) + d(u_{2k}, C_1) \geq t_1 + 2$. By (2) and Lemma 2.3, $d(u_0, C_1) \leq 2$, and therefore $d(u_{2k}, C_1) = t_1$ and $N(u_0, C_1) = \{x_i, x_{i+2}\}$, where $i \in \{2, 2t_1 - 2\}$. Let $C'_1 = C_1 - x_{i+1} + u_0$ and $P' = P + u_{2k}x_{i+1}$. Then $l(C'_1) = l(C_1)$ and $l(P') = l(P) + 1$, contradicting (3). Hence we must have $r(e_3, P) = 1$ and $d(u_0, C_i) + d(u_{2k}, C_i) = t_i + 1$ for each $i \in \{1, 2\}$. It follows that $d(u_0, H) + d(u_{2k}, H) = t + 1$, and so $d(u_0, P) > 0$. By (3), this implies that $G[V(P)]$ does not contain an e_3 -Hamiltonian cycle. Then $u_1 u_{2k} \notin E$, and, by Lemma 2.2, $d(u_1, P) + d(u_{2k}, P) \leq k + 1$. It follows that $d(u_1, C_1 \cup C_2) + d(u_{2k}, C_1 \cup C_2) \geq n + 3 - (k + 1) \geq t_1 + t_2 + 3$. We may assume w.l.o.g. that $d(u_1, C_1) + d(u_{2k}, C_1) \geq t_1 + 2$. By (5), we obtain that $t_1 = 2$ and $d(u_1, C_1) = d(u_{2k}, C_1) = 2$. Let P_1 and P_2 be the two components of $P - e_3$ and say w.l.o.g. that $|E(P_1)| \leq |E(P_2)|$ and $u_1 \in V(P_1)$. Let $C''_1 = C_1 - x_3 + u_1$ and $P'' = P - u_1 + u_{2k}x_3$. Then we have $l(C''_1) = l(C_1)$, $l(P'') = l(P)$, and $\sigma(e_3, P'') = \sigma(e_3, P) - 1$, contradicting (4).

Case b. $s = 2k + 1$. In this case, $u_{2k+1} \in V_1$. Then either $e_3 = u_{2i-1}u_{2i}$ or $e_3 = u_{2i+1}u_{2i}$ for some $i \in \{1, 2, \dots, k\}$. W.l.o.g., say the former holds. Clearly, $r(e_3, P - u_1) = 0$ and $\sigma(e_3, P - u_1) > 0$ if e_3 is on $P - u_1$. Choose u_0 from $V(H) - V(P)$ such that $u_0 \in V_2$. By Lemma 2.1, if $d(u_0, P - u_1) + d(u_{2k+1}, P - u_1) \geq k + 1$, then G has a path P' from u_0 to u_2 such that $V(P') = V(P - u_1) \cup \{u_0\}$, and, moreover, P' is an e_3 -path if e_3 is on $P - u_1$. Thus $P' + u_1 u_2$ is an e_3 -path, contradicting the fact that P is a longest e_3 -path in H . Hence $d(u_0, P) + d(u_{2k+1}, P) = d(u_0, P - u_1) + d(u_{2k+1}, P - u_1) \leq k$. Consequently, $d(u_0, H) + d(u_{2k+1}, H) \leq t - 1$, and so $d(u_0, C_1 \cup C_2) + d(u_{2k+1}, C_1 \cup C_2) \geq n + 3 - (t - 1) = t_1 + t_2 + 4$. By (5), $d(u_0, C_i) \leq 2$ and $d(u_{2k+1}, C_i) \leq 2$ for $i = 1, 2$. It follows that $t_1 = t_2 = 2$ and $d(u_0, C_1) = d(u_{2k+1}, C_1) = 2$. Let $C'_1 = C_1 - x_2 + u_0$ and $P'' = P + u_{2k+1}x_2$. Then $l(C'_1) = l(C_1)$ and $l(P'') = l(P) + 1$, contradicting (3). \square

Claim 5. $\sigma(e_3, P) = 0$.

Proof of Claim 5. On the contrary, suppose $\sigma(e_3, P) > 0$. By Lemma 2.2, we have

$$d(u_1, P) + d(u_{2t}, P) \leq t + r(e_3, P),$$

and it follows that

$$d(u_1, C_1 \cup C_2) + d(u_{2t}, C_1 \cup C_2) \geq t_1 + t_2 + 3 - r(e_3, P).$$

First, suppose that, for some $i \in \{1, 2\}$, say $i = 1$, $d(u_1, C_1) + d(u_{2t}, C_1) \geq t_1 + 2$. By (5), we obtain $t_1 = 2$ and $d(u_1, C_1) = d(u_{2t}, C_1) = 2$. As in the proof of Claim 4, it is easy to see that $G[V(C_1 \cup P)]$ contains an e_1 -cycle C'_1 and an e_3 -path P' such

that $V(C'_1) \cap V(P') = \emptyset$, $l(C'_1) = l(C_1)$, $l(P') = l(P)$, and $\sigma(e_3, P') = \sigma(e_3, P) - 1$, contradicting (4). Therefore we must have

$$(6) \quad r(e_3, P) = 1 \quad \text{and} \quad d(u_1, P) + d(u_{2t}, P) = t + 1,$$

$$(7) \quad d(u_1, C_i) + d(u_{2t}, C_i) = t_i + 1, \quad i = 1, 2.$$

By (6) and the remark of Lemma 2.2, we must have $\{u_1 u_{2a+2}, u_{2t} u_{2a+1}\} \subseteq E$, where $e_3 = u_{2a+1} u_{2a+2}$. Let

$$\begin{aligned} L' &= u_1 u_2 \dots u_{2a}, C' = u_1 u_2 \dots u_{2a+1} u_{2a+2} u_1, \\ L'' &= u_{2a+3} u_{2a+4} \dots u_{2t}, \quad \text{and} \quad C'' = u_{2a+1} u_{2a+2} \dots u_{2t} u_{2a+1}. \end{aligned}$$

Clearly,

$$P'' = u_{2a} u_{2a-1} \dots u_1 u_{2a+2} u_{2a+1} u_{2t} u_{2t-1} \dots u_{2a+3}$$

is an e_3 -Hamiltonian path of H with $\sigma(e_3, P) = \sigma(e_3, P'')$. Similarly, we must have

$$(8) \quad d(u_{2a}, P'') + d(u_{2a+3}, P'') = t + 1,$$

$$(9) \quad d(u_{2a}, C_i) + d(u_{2a+3}, C_i) = t_i + 1, \quad i = 1, 2.$$

If $d(u_1, C_1) + d(u_{2a}, C_1) \geq t_1 + 1$, then, by Lemma 2.4, $G[V(C_1 \cup L')]$ contains an e_1 -Hamiltonian cycle C''_1 , and, consequently, C''_1, C_2, C'' are three required cycles, a contradiction. Hence $d(u_1, C_1) + d(u_{2a}, C_1) \leq t_1$. Similarly, we can show that $d(u_{2a+3}, C_1) + d(u_{2t}, C_1) \leq t_1$. It follows that $\sum_{u \in U} d(u, C_1) \leq 2t_1$, where $U = \{u_1, u_{2a}, u_{2a+3}, u_{2t}\}$. By (7) and (9), $\sum_{u \in U} d(u, C_1) \geq 2t_1 + 2$, a contradiction. \square

By Claim 5, say $e_3 = u_{2t-1} u_{2t}$. As $\delta_2(G) \geq n + 3$, G is connected, so \overline{G}^b is well defined. We divide the proof of the theorem into the following two cases.

Case 1. $t = 1$. As $n \geq 6$, either $t_1 = 3$ or $t_2 = 3$, say $t_2 = 3$. We break into the following two subcases: $t_1 = 2$ or $t_1 = 3$.

Subcase 1.1. $t_1 = 3$. In this subcase, $n = 7$ and $\delta_2(G) \geq 10$. By (2), we see that either $\{x_2, x_6\} \not\subseteq N(y_i)$ for each $i \in \{3, 5\}$ or $\{y_2, y_6\} \not\subseteq N(x_i)$ for each $i \in \{3, 5\}$. W.l.o.g., say the former holds. Let $x' \in \{x_2, x_6\}$ be such that $x' y_3 \notin E$. As $d(x') + d(y_3) \geq 10$, we see that $d(y_3, C_1) = 2$ and $u_2 y_3 \in E$. Similarly, we must have $d(y_5, C_1) = 2$ and $u_2 y_5 \in E$. Therefore we must have $x_4 \in N(y_3) \cap N(y_5)$. By (2), we see that $u_1 y_2 \notin E$ and $u_1 y_4 \notin E$. As $d(u_1) + d(y_j) \geq 10$ for each $j \in \{2, 4\}$, we see that $u_1 y_6 \in E$ and $d(u_1, C_1) = d(y_j, C_1) = 3$ for each $j \in \{2, 4\}$. Then $C'_1 = u_1 u_2 y_3 x_4 u_1$ and $C'_2 = y_1 y_2 x_3 y_4 y_5 y_6 y_1$ are two cycles passing through e_3 and e_2 , respectively, and $l(C'_1) + l(C'_2) < l(C_1) + l(C_2)$, contradicting (2).

Subcase 1.2. $t_1 = 2$. In this subcase, $n = 6$ and $\delta_2(G) \geq 9$. By (2), either $u_2 y_3 \notin E$ or $u_1 y_4 \notin E$, say $u_2 y_3 \notin E$. As $d(y_3) + d(u_2) \geq 9$, we see that $d(u_2, C_1) = d(u_2, C_2) = d(y_3, C_1) = 2$. As $u_2 y_5 \in E$ and by (2), we have $u_1 y_4 \notin E$. Similarly, we must have $d(u_1, C_1) = d(u_1, C_2) = d(y_4, C_1) = 2$. Thus $x_1 x_4 y_3 y_4 x_1$ and $u_1 u_2 x_3 x_2 u_1$ are two cycles passing through e_1 and e_3 , respectively. This contradicts (2).

Case 2. $t \geq 2$. By (2) and Lemma 2.2, we have

$$(10) \quad d(u_1, P) + d(u_{2t}, P) \leq t$$

and so

$$(11) \quad d(u_1, C_1 \cup C_2) + d(u_{2t}, C_1 \cup C_2) \geq n + 3 - t = t_1 + t_2 + 3.$$

This implies that there exists $i \in \{1, 2\}$, say $i = 2$, such that $d(u_1, C_2) + d(u_{2t}, C_2) \geq t_2 + 2$. By (2) and Lemma 2.3, we obtain

$$(12) \quad d(u_{2t}, C_2) = t_2 \text{ and } N(u_1, C_2) = \{y_a, y_{a+2}\} \text{ for some } a \in \{2, 2t_2 - 2\}$$

and it follows that

$$(13) \quad d(u_1, C_1) + d(u_{2t}, C_1) \geq t_1 + 1.$$

We break into the three following subcases according to the values of t_1 and t_2 .

Subcase 2.1. $t_2 = 3$. Let $2b - 1$ be the largest integer in $\{1, 3, \dots, 2t - 3\}$ such that

$$(14) \quad u_i u_{2t} \notin E \text{ and } d(u_i, C_1) = d(u_i, C_2) = 2 \text{ for all } i \in \{1, 3, \dots, 2b - 1\},$$

$$(15) \quad y_{a+1} u_j \notin E, \quad d(u_j, C_1) = 2 \text{ and } d(u_j, C_2) = 1 \text{ for all } j \in \{2, 4, \dots, 2b - 2\},$$

$$(16) \quad N(u_{2t}, C_1) = \{x_1\}, \quad d(y_{a+1}, C_1) = 2 \text{ and } G[\{u_1, u_2, \dots, u_{2b-1}\}] \cong K_{b, b-1}.$$

If there exists no such an integer $2b - 1$ satisfying (14), (15), and (16), let $2b - 1 = 1$.

Since $d(u_1, C_2) = 2$ and $d(u_{2b-1}, C_2) = 2$ by (12) and (14), it is easy to see that $G[V(C_2) \cup \{u_1, u_2, \dots, u_{2b-1}\} - \{y_{a+1}\}]$ contains an e_2 -Hamiltonian cycle C'_2 . Let $L' = u_{2b} u_{2b+1} \dots u_{2t} y_{a+1}$. Then $G[V(L')]$ does not contain an e_3 -Hamiltonian cycle. Thus $y_{a+1} u_{2b} \notin E$. Clearly, $r(e_3, L') = 0$. Then by Lemma 2.2, $d(u_{2b}, L') + d(y_{a+1}, L') \leq t - b + 1$. As $y_{a+1} u_j \notin E$ for all $j \in \{2, 4, \dots, 2b - 2\}$ by (15), we have $d(y_{a+1}, P) + d(u_{2b}, P) \leq t + 1$, and, if the equality holds, then $\{u_1, u_3, \dots, u_{2b-1}\} \subseteq N(u_{2b})$. Suppose $d(u_{2b}, C_2) = 2$. Then either $y_1 y_6 y_5 u_{2b} y_1$ is an e_2 -cycle in G or $y_{a+1} = y_5$ and $\{y_1, y_3\} \subseteq N(u_{2b})$. By (2), the latter holds. Then $u_{2b-1} y_2 \notin E$, for otherwise $C_1, y_1 y_6 y_5 y_4 u_1 u_2 \dots u_{2b-1} y_2 y_1$, and $y_3 u_{2b} u_{2b+1} \dots u_{2t} y_3$ are three required cycles. Hence $\{y_4, y_6\} \subseteq N(u_{2b-1})$. Thus $y_1 y_6 u_{2b-1} u_{2b} y_1$ is an e_2 -cycle in G , contradicting (2). Therefore we must have $d(u_{2b}, C_2) \leq 1$. As C_2 and $C_2 - y_{a+1} + u_1$ are two e_3 -cycles, we see that $d(y_{a+1}, C_1) \leq 2$ and $d(u_{2b}, C_1) \leq 2$ by (2) and Lemma 2.3. As $d(y_{a+1}) + d(u_{2b}) \geq n + 3$, it follows that

$$(17) \quad \begin{aligned} t_1 = 2, \quad d(y_{a+1}, C_1) = d(u_{2b}, C_1) = 2, \quad d(u_{2b}, C_2) = 1, \\ \text{and } \{u_1, u_3, \dots, u_{2b-1}\} \subseteq N(u_{2b}). \end{aligned}$$

If $x_3 u_{2t} \in E$, then $C_1 - x_3 + y_{a+1}, C'_2$ and $x_3 u_{2b} u_{2b+1} \dots u_{2t} x_3$ are three required cycles, a contradiction. Together with (13), we obtain

$$(18) \quad d(u_1, C_1) = 2 \quad \text{and} \quad N(u_{2t}, C_1) = \{x_1\}.$$

This indeed shows $2b - 1 \geq 1$ such that (14), (15), and (16) are satisfied. We claim $2b - 1 = 2t - 3$. If this is not true, i.e., $2b - 1 < 2t - 3$, let $L = u_{2b+1} u_{2b+2} \dots u_{2t}$. By (17) and (18), $d(u_1, C_1) + d(u_{2b}, C_1) = 4$, and, clearly, $G[V(C_1) \cup \{u_1, \dots, u_{2b}\}]$ contains an e_1 -Hamiltonian cycle. Therefore $G[V(L)]$ does not contain an e_3 -Hamiltonian cycle. Hence $u_{2b+1} u_{2t} \notin E$, and, by Lemma 2.2, $d(u_{2b+1}, L) + d(u_{2t}, L) \leq t - b$. As $u_{2t} u_i \notin E$ for all $i \in \{1, 3, \dots, 2b - 1\}$ by (14), we have $d(u_{2b+1}, P) + d(u_{2t}, P) \leq t$, and, if the equality holds, then $\{u_2, u_4, \dots, u_{2b}\} \subseteq N(u_{2b+1})$. By (5), $d(u_{2b+1}, C_1 \cup C_2) \leq 4$. With (18), we conclude that $8 \geq d(u_{2b+1}, C_1 \cup C_2) + d(u_{2t}, C_1 \cup C_2) \geq n + 3 - t = t_1 + t_2 + 3 = 8$. It follows that $d(u_{2b+1}, C_1) = d(u_{2b+1}, C_2) = 2$ and $\{u_2, u_4, \dots, u_{2b}\} \subseteq N(u_{2b+1})$. Together with (17) and (18), this shows that (14), (15), and (16) hold if $2b - 1$ is replaced by $2b + 1$, contradicting the maximality of $2b - 1$. Hence $2b - 1 = 2t - 3$.

By (14)–(17), we see that $d(u_2, C_1) = 2$, $d(u_2, C_2) = 1$ and $G[\{u_1, u_2, \dots, u_{2t-2}\}] \cong K_{t-1, t-1}$. First, suppose $y_1 u_2 \in E$. By (2), for each $i \in \{1, 3, \dots, 2t-3\}$, $y_1 y_6 u_i u_2 y_1$ is not an e_2 -cycle in G and therefore $u_i y_6 \notin E$. As $d(u_1) + d(y_6) \geq n+3$ and $u_1 u_{2t} \notin E$, we see that $d(y_6, C_1) = 2$. As $d(y_{a+1}, C_1) = 2$ by (17), we see that $C_1 - x_3 + y_{a+1}$ and $x_3 u_2 y_1 y_6 x_3$ are two cycles of length 4, contradicting (2).

Thus $y_1 u_2 \notin E$. Similarly, by (14)–(17), we can show that $y_1 u_j \notin E$ for all $j \in \{2, 4, \dots, 2t-2\}$. It follows that $\{u_2, u_4, \dots, u_{2t-2}\} \subseteq N(y_c)$, where $\{y_1, y_{a+1}, y_c\} = \{y_1, y_2, y_3\}$. As $d(y_1) + d(u_2) \geq n+3$, we see that $u_2 u_{2t-1} \in E$. Then $y_c u_2 u_{2t-1} u_{2t} y_c$ is an e_3 -cycle in G , contradicting (2).

Subcase 2.2. $t_2 = 2$ and $t_1 = 3$. By Subcase 2.1 and symmetry, we may assume $d(u_1, C_1) + d(u_{2t}, C_1) \leq t_1 + 1$. It follows from (10), (11), and (13) that

$$(19) \quad d(u_1, P) + d(u_{2t}, P) = t \quad \text{and} \quad d(u_1, C_1) + d(u_{2t}, C_1) = 4.$$

Let $C'_2 = C_2 - y_3 + u_1$. Clearly, C'_2 is an e_2 -cycle in G . Therefore $G[V(L')]$ does not contain an e_3 -Hamiltonian cycle where $L' = u_2 u_3 \dots u_{2t} y_3$. Then $y_3 u_2 \notin E$. As $r(e_3, L') = 0$ and by Lemma 2.2, we obtain $d(y_3, L') + d(u_2, L') \leq t$. It follows that $d(u_2, C_2 \cup P) + d(y_3, C_2 \cup P) \leq t + 4$ and, if the equality holds, then $y_1 u_2 \in E$. Thus $d(u_2, C_1) + d(y_3, C_1) \geq n + 3 - t - 4 = 4$. As C_2 and C'_2 are two e_2 -cycles of length 4, we see, by (2) and Lemma 2.3, that $d(u_2, C_1) \leq 2$ and $d(y_3, C_1) \leq 2$. We conclude that

$$(20) \quad d(y_3, C_1) = 2, \quad d(u_2, C_1) = 2, \quad \text{and} \quad y_1 u_2 \in E.$$

Let $C''_2 = y_1 y_4 u_1 u_2 y_1$. We have

$$(21) \quad y_2 u_3 \notin E,$$

for otherwise C_1, C''_2 , and $y_3 y_2 u_3 u_4 \dots u_{2t} y_3$ are three required cycles.

We claim $t = 2$. On the contrary, suppose $t \geq 3$. Since $y_1 u_2 u_1 y_2 y_3 y_4 y_1$ is an e_2 -cycle in G , $G[V(L'')]$ does not contain an e_3 -Hamiltonian cycle, where $L'' = u_3 u_4 \dots u_{2t}$. So $u_3 u_{2t} \notin E$ and $d(u_3, L'') + d(u_{2t}, L'') \leq t - 1$ by Lemma 2.2. By (5), $d(u_3, C_1) \leq 2$. As $d(u_3) + d(u_{2t}) \geq n + 3$, we see, together with (21), that $d(u_3, C_1) = 2$ and $y_4 u_3 \in E$. Then $y_1 y_2 u_1 u_2 u_3 y_4 y_1$ is an e_2 -cycle in G . Thus $G[V(L)]$ does not contain an e_3 -Hamiltonian cycle where $L = u_4 u_5 \dots u_{2t} y_3$, and so $y_3 u_4 \notin E$. As $r(e_3, L) = 0$ and by Lemma 2.2, $d(u_4, L) + d(y_3, L) \leq t - 1$. By (5), $d(u_4, C_1) \leq 2$. As $d(u_4) + d(y_3) \geq n + 3$, it follows that $u_1 u_4 \in E$, and, consequently, $C_1, y_1 y_4 u_3 u_2 y_1$, and $y_3 y_2 u_1 u_4 u_5 \dots u_{2t} y_3$ are three required cycles, a contradiction. This shows that $t = 2$.

Thus $n = 7$ and $\delta_2(G) \geq 10$. Since C_2, C'_2 , and C''_2 are e_2 -cycles in G and by (2), we see that

$$(22) \quad \begin{aligned} \{x_1, x_5\} \not\subseteq N(u_2), \quad \{x_1, x_5\} \not\subseteq N(y_2), \\ \{x_2, x_6\} \not\subseteq N(y_3), \quad \text{and} \quad \{x_2, x_6\} \not\subseteq N(u_1). \end{aligned}$$

By (2) and (20), we see that $x_3 u_2 \in E$ and $y_3 x_4 \in E$. Then $x_3 u_4 \notin E$, for otherwise $x_3 u_2 u_3 u_4 x_3$ is an e_3 -cycle in G , contradicting (2). As $d(x_3) + d(u_4) \geq 10$, we obtain $\{x_1, x_5\} \subseteq N(u_4)$ and $\{y_2, y_4\} \subseteq N(x_3)$. Then $x_4 u_3 \notin E$, for otherwise $x_4 u_3 u_4 x_5 x_4$ is an e_3 -cycle in G , contradicting (2). As $d(x_4) + d(u_3) \geq 10$ and $y_2 u_3 \notin E$ by (21), we obtain $\{u_1, y_1\} \subseteq N(x_4)$ and $\{x_2, x_6, y_4\} \subseteq N(u_3)$. Then $y_3 x_2 \notin E$, for otherwise $y_3 x_2 u_3 u_4 y_3$ and C'_2 are two cycles of length 4 in G , contradicting (2). As

$d(x_2) + d(y_3) \geq 10$, we obtain $\{y_1, u_1\} \subseteq N(x_2)$ and $x_6y_3 \in E$. We have $x_5u_2 \notin E$, for otherwise $x_5u_2u_3u_4x_5$ is an e_3 -cycle in G , contradicting (2). As $d(x_5) + d(u_2) \geq 10$, we obtain $\{y_2, y_4\} \subseteq N(x_5)$ and $x_1u_2 \in E$. We also see $x_1y_2 \notin E$, for otherwise C_2'' and $x_1y_2x_5x_6x_1$ are two cycles of length 4 in G , contradicting (2). Then $x_1y_4 \in E$ as $d(x_1) + d(y_2) \geq 10$. Finally, we have $u_1x_6 \notin E$, for otherwise $x_1x_6u_1x_2x_1$ is an e_1 -cycle in G , contradicting (2), and so $x_6y_1 \in E$ as $d(u_1) + d(x_6) \geq 10$. Then \overline{G}^b consists of three cycles $x_1y_2u_3x_4x_1$, $x_3x_6u_1u_4x_3$, $x_5x_2y_3u_2x_5$ and two isolated vertices y_1 and y_4 . Therefore $G \cong G_3$.

Subcase 2.3. $t_1 = t_2 = 2$. In this subcase, by (11), we have

$$(23) \quad d(u_1, C_1 \cup C_2) \geq 3 \quad \text{and} \quad d(u_1, C_1) > 0.$$

Let $C_2' = C_2 - y_3 + u_1$ and $L_1 = P - u_1 + u_{2t}y_3$. As before, $G[V(L_1)]$ does not contain an e_3 -Hamiltonian cycle, and therefore $y_3u_2 \notin E$ and $d(y_3, L_1) + d(u_2, L_1) \leq t$. Hence

$$(24) \quad d(y_3, C_1 \cup C_2) + d(u_2, C_1 \cup C_2) \geq n + 3 - t - 1 \geq 6.$$

It follows that

$$(25) \quad d(u_2, C_1 \cup C_2) \geq 2 \quad \text{and} \quad d(y_3, C_1) \geq 1.$$

We divide the proof into the following two subcases: $t \geq 3$ or $t = 2$.

Subcase 2.3(a). $t \geq 3$. Let $L_2 = u_3u_4 \dots u_{2t}$. As $d(u_1, C_2) = 2$ and by (23), (25), and Lemma 2.4, if $d(u_2, C_1) = 2$, then $G[V(C_1) \cup \{u_1, u_2\}]$ contains an e_1 -Hamiltonian cycle as $d(u_1, C_1) > 0$, and otherwise $d(u_2, C_2) = 1$ and $G[V(C_2) \cup \{u_1, u_2\}]$ contains an e_2 -Hamiltonian cycle. Therefore $G[V(L_2)]$ does not contain an e_3 -Hamiltonian cycle. Therefore $u_3u_{2t} \notin E$ and $d(u_3, L_2) + d(u_{2t}, L_2) \leq t - 1$ by Lemma 2.2. Hence $d(u_3, P) + d(u_{2t}, P) \leq t$. It follows that

$$(26) \quad d(u_3, C_1 \cup C_2) + d(u_{2t}, C_1 \cup C_2) \geq n + 3 - t = 7 \quad \text{and} \quad d(u_3, C_1 \cup C_2) \geq 3.$$

As $d(u_3, C_2) > 0$ by (26) and $d(u_1, C_2) = 2$, it is easy to see that $G[V(C_2) \cup \{u_1, u_2, u_3\} - \{y_3\}]$ contains an e_2 -Hamiltonian cycle Q . Then $G[V(L_3)]$ does not contain an e_3 -Hamiltonian cycle where $L_3 = u_4u_5 \dots u_{2t}y_3$. Hence $y_3u_4 \notin E$. As $r(e_3, L_3) = 0$ and by Lemma 2.2, we have $d(u_4, L_3) + d(y_3, L_3) \leq t - 1$. Therefore $d(u_4, P) + d(y_3, P) \leq t + 1$, and, moreover, if the equality holds, then $u_1u_4 \in E$. We now claim $u_1u_4 \in E$. On the contrary, say $u_1u_4 \notin E$. Then $d(u_4, C_1 \cup C_2) + d(y_3, C_1 \cup C_2) \geq n + 3 - t = 7$. This implies that $d(u_4, C_1) = d(y_3, C_1) = 2$ and $y_1u_4 \in E$. If $x_3u_{2t} \in E$, then $C_1 - x_3 + y_3$, Q , and $x_3u_4u_5 \dots u_{2t}x_3$ are three required cycles, a contradiction. Hence $x_3u_{2t} \notin E$. By (13) and (26), we have $d(u_1, C_1) = d(u_3, C_1) = d(u_3, C_2) = 2$. If $x_1u_2 \in E$, then $x_1x_4u_1u_2x_1$, $C_2 - y_3 + u_3$, and $x_3x_2y_3u_{2t}u_{2t-1} \dots u_4x_3$ are three required cycles, a contradiction. Thus $x_1u_2 \notin E$. As $u_2y_3 \notin E$ and by (25), we have $\{u_2x_3, u_2y_1\} \subseteq E$. Let $C_1' = C_1 - x_3 + u_1$ and $L_4 = P - u_1 + x_3u_2$. Then $G[V(L_4)]$ does not contain an e_3 -Hamiltonian cycle. By Lemma 2.2, $d(x_3, L_4) + d(u_{2t}, L_4) \leq t$. As $d(x_3) + d(u_{2t}) \geq n + 3$, we obtain $d(x_3, C_2) = 2$. Then $C_1 - x_3 + u_3$, $y_1y_4u_1u_2y_1$ and $x_3u_4u_5 \dots u_{2t}y_3y_2x_3$ are three required cycles, a contradiction. This shows $u_1u_4 \in E$.

Then we have

$$(27) \quad d(u_4, C_1 \cup C_2) + d(y_3, C_1 \cup C_2) \geq n + 3 - t - 1 = 6.$$

We shall obtain a contradiction according to whether $u_2y_1 \in E$ or $u_2y_1 \notin E$. Let us first assume that $u_2y_1 \in E$. Let $C_2'' = y_1y_4u_1u_2y_1$. Then $G[V(L_5)]$ does not contain an e_3 -Hamiltonian cycle where $L_5 = u_3u_4 \dots u_{2t}y_3y_2$. Hence $y_2u_3 \notin E$. By (26), we obtain $d(u_3, C_1) = d(u_{2t}, C_1) = 2$ and $u_3y_4 \in E$. Then $x_3u_2 \notin E$, for otherwise $C_1 - x_3 + u_3$, C_2 and $x_3u_2u_1u_4u_5 \dots u_{2t}x_3$ are three required cycles. By (24), we obtain $d(y_3, C_1) = 2$ and $x_1u_2 \in E$. We also have $x_3u_4 \notin E$, for otherwise $C_1 - x_3 + u_3$, $y_1u_2u_1y_2y_3y_4y_1$, and $x_3u_4 \dots u_{2t}x_3$ are three required cycles. As $y_3u_4 \notin E$ and by (27), $\{x_1u_4, y_1u_4\} \subseteq E$. If $u_1x_4 \in E$, then $x_1x_4u_1u_2x_1$, C_2 and $x_2x_3u_{2t}u_{2t-1} \dots u_3x_2$ are three required cycles. Hence $u_1x_4 \notin E$ and so $u_1x_2 \in E$ by (23). Then $x_1x_4u_3u_2x_1$, C_2 , and $x_2x_3u_{2t}u_{2t-1} \dots u_4u_1x_2$ are three required cycles, again a contradiction.

Next, we assume $u_2y_1 \notin E$. As $y_3u_2 \notin E$ and by (24), we have $d(u_2, C_1) = d(y_3, C_1) = 2$. Then $x_3u_{2t} \notin E$, for otherwise $C_1 - x_3 + y_3$, C_2' and $x_3u_2u_3 \dots u_{2t}x_3$ are three required cycles. By (13) and (26), we have $d(u_1, C_1) = d(u_3, C_1) = 2$. Then $C_1 - x_2 + u_2$, C_2' , and $y_3x_2u_3u_4 \dots u_{2t}y_3$ are three required cycles, a contradiction.

Subcase 2.3(b). $t = 2$. In this subcase, $n = 6$ and $\delta_2(G) \geq 9$. As $\delta_2(G) \geq 9$, each component of \overline{G}^b is a path of length at most 2. If $G \not\geq G_1$, then $\overline{G}_1^b \not\geq \overline{G}^b$. Then \overline{G}^b must consist of three paths of length 2 and three isolated vertices such that the three isolated vertices are all in V_1 or are all in V_2 . This would imply $G \cong G_0$. Therefore we may assume that $G \geq G_1$. Hence we need to show $G \leq G_2$.

As $y_3u_2 \notin E$ and by (25), we have either $y_1u_2 \in E$ or $N(u_2, C_1 \cup C_2) = \{x_1, x_3\}$.

Subcase 2.3(b1). $y_1u_2 \in E$. Let $C_2'' = y_1y_4u_1u_2y_1$. Then $u_3y_2 \notin E$, for otherwise C_1, C_2'' , and $y_2u_3u_4y_3y_2$ are three required cycles. Thus $E(\overline{G}^b) \supseteq \{u_1u_4, y_3u_2, y_2u_3\}$. Similarly, if $d(u_1, C_1) = d(u_4, C_1) = 2$, then $u_2x_3 \notin E$; otherwise the cycles $C_1 - x_3 + u_1$, C_2 and $P - u_1 + x_3$ are as required. This means $u_2x_1 \in E$, by (25), and so $u_3x_2 \notin E$; otherwise the cycles $u_2x_1x_4u_1u_2, C_2$, and $x_2u_3u_4x_3x_2$ are as required. Hence $E(\overline{G}^b) \supseteq \{u_2x_3, u_3x_2\}$, and so $G \leq G_2$. Therefore, by (13), we may assume that $d(u_1, C_1) + d(u_4, C_1) = 3$.

First, suppose $d(u_1, C_1) = 2$ and $d(u_4, C_1) = 1$. Thus $x_iu_4 \notin E$ for some $i \in \{1, 3\}$. Assume that $G \not\geq G_2$. Then we must have $d(y_3, C_1) = d(u_3, C_1) = 2$ and $u_3y_4 \in E$. Suppose $x_3u_4 \in E$. Then $x_3u_2 \notin E$, for otherwise $C_1 - x_3 + u_1$, C_2 , and $P - u_1 + x_3$ are three required cycles. Consequently, $x_1u_2 \in E$ as $y_3u_2 \notin E$ and by (25), and so $x_1u_2u_1x_4x_1, C_2$ and $x_3u_4u_3x_2x_3$ are three required cycles, a contradiction. Hence $x_3u_4 \notin E$ and so $x_1u_4 \in E$. Then $\{y_2, y_4, u_2\} \subseteq N(x_3)$ as $d(x_3) + d(u_4) \geq 9$. Thus $u_2x_1 \notin E$, for otherwise $x_1x_4u_1u_2x_1, C_2 - y_3 + x_3$ and $y_3u_4u_3x_2y_3$ are three required cycles, and therefore $\{y_2, y_4\} \subseteq N(x_1)$ as $d(u_2) + d(x_1) \geq 9$. It follows that $x_1x_4u_1y_2x_1, x_3y_4y_1u_2x_3$, and $y_3u_4u_3x_2y_3$ are three required cycles, a contradiction.

Next, suppose $d(u_1, C_1) = 1$ and $d(u_4, C_1) = 2$. Thus $x_iu_1 \notin E$ for some $i \in \{2, 4\}$. Assume $G \not\geq G_2$. Then we must have $d(y_2, C_1) = d(u_2, C_1) = 2$. As $d(x_i) + d(u_1) \geq 9$, we see that $\{y_1, y_3, u_3\} \subseteq N(x_i)$. If $i = 2$, i.e., $x_4u_1 \in E$, then $x_1x_4u_1u_2x_1, C_2$, and $x_3x_2u_3u_4x_3$ are three required cycles, a contradiction. Therefore $x_4u_1 \notin E$ and $x_2u_1 \in E$. Then $x_2y_3 \notin E$, for otherwise $C_1 - x_3 + y_3, C_2'$, and $P - u_1 + x_3$ are three required cycles. As $d(x_2) + d(y_3) \geq 9$, we see that $x_2u_3 \in E$. Then $x_1x_4y_3y_2x_1, C_2''$, and $x_3x_2u_3u_4x_3$ are three required cycles, a contradiction.

Subcase 2.3(b2). $N(u_2, C_1 \cup C_2) = \{x_1, x_3\}$. As $\delta_2(G) \geq 9$, we see that $N(y_1) = N(y_3) = V_2 - \{u_2\}$. Then $x_3u_4 \notin E$, for otherwise $C_1 - x_3 + y_3, C_2'$, and $P - u_1 + x_3$ are three required cycles. Thus $d(u_1, C_1) = 2$ by (13). As $d(x_3) + d(u_4) \geq 9$, we see that $d(x_3, C_2) = 2$. Assume that $G \not\geq G_2$. Then we see that either $u_3y_2 \in E$ or $u_3x_2 \in E$. If $u_3y_2 \in E$, then $x_1x_4u_1u_2x_1, x_3x_2y_1y_4x_3$, and $y_3y_2u_3u_4y_3$ are three required cycles,

and if $u_3x_2 \in E$, then $C_1 - x_2 + u_2$, C'_2 and $x_2y_3u_4u_3x_2$ are three required cycles, a contradiction. This completes the proof of the theorem.

REFERENCES

- [1] C. BERGE, *Graphs*, North-Holland, Amsterdam, 1985, pp. 200–217.
- [2] B. BOLLOBÁS, *Extremal Graph Theory*, Academic Press, London, 1978.
- [3] J. A. BONDY AND V. CHVÁTAL, *A method in graph theory*, Discrete Math., 15 (1976), pp. 111–135.
- [4] O. ORE, *Note on Hamilton circuits*, Amer. Math. Monthly, 67 (1960), p. 55.
- [5] H. WANG, *Covering a bipartite graph with cycles passing through given edges*, Australas. J. Combin., 19 (1999), pp. 115–121.

PRACTICAL APPROXIMATION ALGORITHMS FOR ZERO- AND BOUNDED-SKEW TREES*

ALEXANDER Z. ZELIKOVSKY[†] AND ION I. MĂNDOIU[‡]

Abstract. The *skew* of an edge-weighted rooted tree is the maximum difference between any two root-to-leaf path weights. Zero- or bounded-skew trees are needed for achieving synchronization in many applications, including network multicasting [G. N. Rouskas and I. Baldine, *IEEE J. on Selected Areas in Communication*, 15 (1997), pp. 346–356] and VLSI clock routing [H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990, A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, Norwell, MA, 1995]. In these applications edge weights represent propagation delays, and a signal generated at the root should be received by multiple recipients located at the leaves (almost) simultaneously. The objective is to find zero- or bounded-skew trees of minimum total weight, since the weight of the tree is directly proportional to the amount of resources (bandwidth and buffers for network multicasting, power and chip area for clock routing in VLSI) that must be allocated to the tree. Charikar et al. in [Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, ACM, New York, 1999, pp. 177–184] have recently proposed the first strongly polynomial algorithms with proven constant approximation factors, $2e \approx 5.44$ and 16.86, for finding minimum weight zero- and bounded-skew trees, respectively.

In this paper we introduce a new approach to these problems, based on zero-skew “stretching” of spanning trees, and obtain algorithms with improved approximation factors of 4 and 14. For the case when tree nodes are points in the plane and edge weights are given by the rectilinear metric our algorithms find zero- and bounded-skew trees of length at most 3 and 9 times the optimum. This case is of special interest in VLSI clock routing. An important feature of our algorithms is their practical running time, which is asymptotically the same as the time needed for computing the minimum spanning tree.

Key words. Steiner trees, clock routing, VLSI physical design, approximation algorithms

AMS subject classifications. 05C05, 05C85, 68W25, 68W35, 68W40

PII. S0895480100378367

1. Introduction. The *skew* of an edge-weighted rooted tree is the maximum difference between any two root-to-leaf path weights. Zero- or bounded-skew trees are needed for achieving synchronization in many applications, including network multicasting [20] and VLSI clock routing [2, 17]. In these applications edge weights represent propagation delays, and a signal generated at the root should be received by multiple recipients, referred to as *sinks*, located at the leaves (almost) simultaneously. The goal is to find zero- or bounded-skew trees of minimum total weight, since the weight of the tree is directly proportional to the amount of resources (bandwidth and buffers for network multicasting, power and chip area for clock routing in VLSI) that must be allocated to the tree.

*Received by the editors September 18, 2000; accepted for publication (in revised form) December 10, 2001; published electronically January 16, 2002. A preliminary version of this work appeared in [22].

<http://www.siam.org/journals/sidma/15-1/37836.html>

[†]Department of Computer Science, Georgia State University, University Plaza, Atlanta, GA 30303 (alexz@cs.gsu.edu). This author’s research was supported in part by NSF grant CCR-9988331, award MM2-3018 of the Moldovan Research and Development Association (MRDA) and the U.S. Civilian Research and Development Foundation for the Independent States of the former Soviet Union (CRDF), and State of Georgia’s Yamacraw Initiative.

[‡]Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114 (mandoiu@cs.ucsd.edu).

In order to meet the skew constraints in the above applications, one may increase edge weights of the underlying network or metric space. This corresponds to adding buffers to a network link, or wire wiggling, respectively. We will refer to this operation as *stretching*. Formally, let (M, d) be an arbitrary metric space. A *stretched tree* $T = (V, E, \pi, cost)$ for a set of sinks $S \subseteq M$ is a rooted tree with node set V and edge set E , together with a pair of mappings, $\pi : V \rightarrow M$ and $cost : E \rightarrow \mathbb{R}_+$, such that

- (1) π is a one-to-one mapping between the leaves of T and S , and
- (2) for every edge $(u, v) \in E$, $cost(u, v) \geq d(\pi(u), \pi(v))$.

Informally, every edge (u, v) of a stretched tree T embedded in (M, d) can be stretched by wiggling such that its length increases from $d(\pi(u), \pi(v))$ to $cost(u, v)$.

A stretched tree T is a *zero-skew tree* (ZST) if all root-to-leaf paths in T have equal cost; T is a *b-bounded-skew tree* (*b*-BST, or just BST when the bound b is clear from the context) if the difference between the cost of any two root-to-leaf paths is at most b .

The two problems that we study in this paper are the following:

Zero-skew tree problem. Given a set of sinks S in metric space (M, d) , find a minimum cost zero-skew tree for S .

Bounded-skew tree problem. Given a set of sinks S in metric space (M, d) and a bound $b > 0$, find a minimum cost b -bounded-skew tree for S .

The ZST and BST problems are NP-hard [8]. The restriction of the BST problem to the rectilinear plane is also known to be NP-hard, but the complexity of the rectilinear ZST problem is not known—for a fixed tree topology the problem can be solved in linear time by using the *deferred-merge embedding* (DME) algorithm independently introduced in [5, 6, 10].

Although the rectilinear zero- and bounded-skew tree problems have received much attention in the VLSI CAD literature [3, 5, 6, 7, 9, 10, 11, 15, 16, 19] (see Chapter 4 of [17] for a detailed review), the first algorithms with constant approximation factors have been proposed only recently by Charikar et al. [8]. They give algorithms with approximation factors of $2e \approx 5.44$ and 16.86 for the ZST and BST problems, respectively. The BST algorithm in [8] relies on an approximation algorithm for the Steiner tree problem in graphs. Using the best current Steiner tree approximation of Robins and Zelikovsky [21] and Arora’s PTAS for computing rectilinear Steiner trees [1], the BST bounds in [8] can be updated to 16.11 for arbitrary metric spaces and to 12.53 for the rectilinear plane (see Table 1).

In this paper we introduce a new approach to these problems, based on zero-skew “stretching” of spanning trees. Our contributions include the following:

- constructive lower bounds on the cost of the optimum ZST and BST in arbitrary metric spaces;
- improved approximation for the ZST problem in arbitrary metric spaces, based on a reduction to the *zero-skew spanning tree problem*;
- improved approximation for the ZST problem in metrically convex metric spaces,¹ based on skew elimination using Steiner points;
- improved approximation for the BST problem in arbitrary and metrically convex metric spaces, based on combining an approximate ZST with a minimum spanning tree for the sinks.

An important feature of our algorithms is their practical running time, which is asymptotically the same as the time needed for computing a minimum spanning tree. Thus,

¹A metric space (M, d) is called *metrically convex* if, for every $u, v \in M$ and $0 \leq \lambda \leq 1$, there exists a point $w \in M$ such that $d(u, w) = \lambda d(u, v)$ and $d(w, v) = (1 - \lambda)d(u, v)$.

TABLE 1

Summary of results and comparison to results of Charikar et al. [8]. Values marked with asterisks update those reported in [8] by taking into account the best current Steiner tree approximation of Robins and Zelikovskiy [21] and Arora’s PTAS for computing rectilinear Steiner trees [1].

Problem	Zero-skew tree			Bounded-skew tree		
	General	M. convex	Rectilinear	General	M. convex	Rectilinear
Approximation factor in [8]	$2e \approx 5.44$			16.11*		12.53*
Approximation factor in this paper	4	3		14	11	9
Runtime in [8]	strongly polynomial			strongly polynomial		
Runtime in this paper	$O(n^2)$		$O(n \log n)$	$O(n^2)$		$O(n \log n)$

our algorithms can easily handle the clock nets with hundreds of thousands of sinks that occur in large cell based or multichip module designs. For a summary of our results and a comparison to the results of Charikar et al. [8]² we refer the reader to Table 1.

The rest of the paper is organized as follows. In the next section we prove new lower bounds on the cost of the optimal ZST and BST. Then, in section 3, we show how to convert (or “stretch”) a rooted tree T spanning the set S of sinks into a ZST for S . We show that such “stretching” increases the cost by the sum of sink delays, where the *delay* in T of a sink s is the length of the path connecting s to its furthest descendant. We also show that, for metrically convex metric spaces such as the Euclidean or rectilinear planes, it is possible to reduce the cost increase to half the sum of delays.

In section 4 we give a Kruskal-like algorithm that builds a rooted spanning tree T whose total delay does not exceed its length and whose length is at most twice the cost of an optimal ZST. These two facts yield an approximation factor of 4 for the ZST problem in arbitrary metric spaces and an approximation factor of 3 for metrically convex metric spaces. In section 5 we discuss the implications of combining our ZST heuristics with the DME algorithm when solving rectilinear instances.

Finally, in section 6, we describe how to construct approximate BSTs by combining an approximate ZST for a subset of the sinks with subtrees of a minimum spanning tree (MST) or approximate minimum Steiner tree for the sinks. In combination with the MST, this gives a 14-approximation algorithm for the BST problem in arbitrary metric spaces; the factor is reduced to 11 for arbitrary metrically convex metric spaces and to 9 for the rectilinear plane.

2. Constructive lower bounds. In this section, we establish new lower bounds for the ZST and BST problems in an arbitrary metric space. In contrast to the lower bounds of Charikar et al. [8] these bounds are constructive. A practical advantage of constructive lower bounds is that they can give tighter bounds on the quality of the computed solution on an instance by instance basis.

The minimum cost of a ZST (BST) for S will be denoted by $ZST^*(S)$ (respectively, $BST^*(S)$). In our analysis we will use the following constructive lower bound on $ZST^*(S)$.

²The running time in [8] is not explicitly estimated.

LEMMA 2.1. *Let S be a set of n sinks. Then, for any enumeration s_1, s_2, \dots, s_n of the sinks in S ,*

$$ZST^*(S) \geq \text{MinDist}\{s_1, s_2\} + \frac{1}{2} \sum_{i=2}^{n-1} \text{MinDist}\{s_1, \dots, s_{i+1}\},$$

where $\text{MinDist}\{A\} = \min_{u, v \in A, u \neq v} d(u, v)$.

Proof. For any $r \geq 0$, let $N(r)$ denote the minimum number of closed balls of radius r of (M, d) needed to cover all sinks in S . Charikar et al. [8] established that

$$ZST^*(S) \geq \int_0^R N(r) dr,$$

where R is the smallest radius r for which $N(r) = 1$.

Let $r_i = \text{MinDist}\{s_1, \dots, s_{i+1}\}/2$ for every $i = 1, \dots, n-1$, and let $r_n = 0$. Clearly, $R \geq r_1 \geq r_2 \geq \dots \geq r_{n-1} \geq r_n$. Note that $N(r) \geq i+1$ for every $r < r_i$, since no two points in the set $\{s_1, \dots, s_{i+1}\}$ can be covered by the same ball of radius r . Hence,

$$\int_0^R N(r) dr \geq \sum_{i=1}^{n-1} \int_{r_{i+1}}^{r_i} (i+1) dr = \sum_{i=1}^{n-1} (i+1)(r_i - r_{i+1}) = 2r_1 + \sum_{i=2}^{n-1} r_i,$$

and the lemma follows. \square

It can be shown that natural greedy enumerations (e.g., start from a diametrical pair of points and add each time the point maximizing minimum distance to previously enumerated points) do not always deliver the maximum to the lower bound established in Lemma 2.1. The complexity of finding the best enumeration is an open question.

Below we bound the cost of the optimum BST by comparing it with the cost of the optimum ZST for a subset of the sinks.

LEMMA 2.2. *Let S be a set of sinks. Then, for any $W \subseteq S$ and skew bound $b > 0$,*

$$BST^*(S) \geq ZST^*(W) - b \cdot (|W| - 1).$$

Proof. Let T be a b -BST for S . We use T to construct a ZST for W of cost no larger than $\text{cost}(T) + b \cdot (|W| - 1)$ as follows. First, notice that T contains a b -BST for W , say T' , as a subtree. Let P_u denote the unique path in T' connecting u to the root, and let u_0 be a leaf of T' for which $\text{cost}(P_{u_0})$ is maximum. We get a ZST for W by adding to T' a loop of cost $\text{cost}(P_{u_0}) - \text{cost}(P_u)$ for each leaf $u \neq u_0$. Since T' has skew at most b , each of the $|W| - 1$ added loops has cost at most b . Thus, the resulting ZST has cost at most $\text{cost}(T') + b \cdot (|W| - 1) \leq BST^*(S) + b \cdot (|W| - 1)$. \square

3. Zero-skew stretching of spanning trees. Let $T = (S, E)$ be a rooted tree spanning a set S of sinks from metric space (M, d) . For any sink u , let T_u denote the subtree of T rooted at u . The *delay* in T of u is defined by

$$\text{delay}_T(u) = \max\{\text{length}(P_{uv}) \mid v \text{ leaf in } T_u\},$$

where P_{uv} denotes the unique path in T connecting u and v , and $\text{length}(P_{uv}) = \sum_{e \in P_{u,v}} d(e)$.

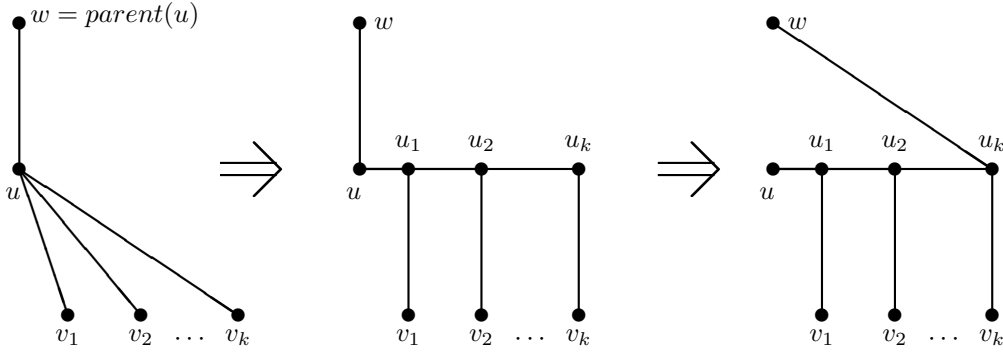


FIG. 1. The two phases of the stretching algorithm for arbitrary metric spaces. In the first phase, for each sink u , $k = \text{deg}_T(u)$ new nodes u_1, \dots, u_k are embedded at u and connected to u by a path of total cost $\text{delay}_T(u)$. The children v_i , $i = 1, \dots, k$, are reattached to the new nodes in nondecreasing order of $d(u, v_i) + \text{delay}_T(v_i)$. In the second phase, the parent of each sink u is reattached to u_k .

Let $\text{length}(T) = \sum_{e \in E} d(e)$ and $\text{delay}(T) = \sum_{u \in S} \text{delay}_T(u)$. In this section we show that, for any metric space (M, d) , T can be stretched to a ZST of cost $\text{length}(T) + \text{delay}(T)$. The stretched ZST uses no Steiner points, i.e., has all nodes embedded at the sinks. We also show that, by using Steiner points, the amount of stretching can be reduced to half the delay of T in case the underlying space is metrically convex.

3.1. Zero-skew stretching in arbitrary metric spaces. The stretching algorithm for arbitrary metric spaces (Figure 2) constructs a ZST T_1 from a given rooted tree T spanning S .³ The construction proceeds in two phases. In the first phase (Steps 1–3) the following transformation is applied to each sink u (see Figure 1). First, the children v_1, \dots, v_k of u are sorted in nondecreasing order of $d(u, v_i) + \text{delay}_T(v_i)$. Then k new nodes u_1, \dots, u_k are embedded at u and connected to u by a path of total cost $\text{delay}_T(u)$. Finally, each v_i is disconnected from u and reattached to u_i by an edge of cost $d(u, v_i)$. The result of the first phase is a tree T_1 in which every sink either is a leaf or has a single child.

In the second phase (Steps 4–5) we convert T_1 into a ZST for S as follows. First, we change the root of T_1 to $r' = r_t$, where r is the root of T and $t = \text{deg}_T(r)$. Notice that every sink u that is not yet a leaf in T_1 is incident to its parent, say v , and to u_1 . For every such sink u the edge (u, v) is replaced in T_1 with (u_k, v) , where $k = \text{deg}_T(u)$. After this transformation all sinks become leaves in T_1 .

LEMMA 3.1. *The stretched tree T_1 produced by the algorithm in Figure 2 is a ZST with total cost $\text{length}(T) + \text{delay}(T)$.*

Proof. We will prove that every path in T_1 from u_k , $u \in S$, $k = \text{deg}_T(u)$, to a descendant sink has cost equal to $\text{delay}_T(u)$; this immediately implies that T_1 is a ZST. Let v_1, \dots, v_k be the sorted children of u in T , and let u_1, \dots, u_k be the copies of u added to T_1 in Step 3. Consider a path P from u_k to a descendant sink s going through edge (u_i, w) , where w is the $\text{deg}_T(v_i)$ th copy of v_i . Inductively, we can assume that the cost of the path from w to s is equal to $\text{delay}_T(v_i)$. Hence, it suffices to show that the cost of the path from u_k to w is equal to $\text{delay}_T(u) - \text{delay}_T(v_i)$. Indeed,

³For clarity, in Figure 2 we omit curly braces for single element sets and use “−” and “+” instead of “\” and “∪”, respectively.

Input: Spanning tree $T = (S, E)$, rooted at r , in a metric space (M, d) .

Output: ZST $T_1 = (V_1, E_1, \pi, cost)$ for S .

-
1. $V_1 \leftarrow S$; $\pi(v) \leftarrow v$ for each $v \in V_1$.
 2. $E_1 \leftarrow E$; $cost(u, v) \leftarrow d(u, v)$ for each $(u, v) \in E_1$.
 3. For each sink $u \in S$, do:
 - $k \leftarrow \deg_T(u)$
 - Sort u 's children in T , say v_1, v_2, \dots, v_k , such that

$$d(u, v_1) + delay_T(v_1) \leq d(u, v_2) + delay_T(v_2) \leq \dots \leq d(u, v_k) + delay_T(v_k)$$
 - // Add k new nodes embedded at u
 - $V_1 \leftarrow V_1 + \{u_1, \dots, u_k\}$; $\pi(u_1) \leftarrow \dots \leftarrow \pi(u_k) \leftarrow u$
 - // Connect the k new nodes and u with a path
 - $E_1 \leftarrow E_1 + (u, u_1)$; $cost(u, u_1) \leftarrow d(u, v_1) + delay_T(v_1)$
 - For $i = 1, \dots, k-1$ do
 - $E_1 \leftarrow E_1 + (u_i, u_{i+1})$
 - $cost(u_i, u_{i+1}) \leftarrow [d(u, v_{i+1}) + delay_T(v_{i+1})] - [d(u, v_i) + delay_T(v_i)]$
 - // Reattach children v_i to the corresponding copies of u
 - For $i = 1, \dots, k$ do
 - $E_1 \leftarrow E_1 - (u, v_i) + (u_i, v_i)$; $cost(u_i, v_i) \leftarrow cost(u, v_i)$.
 4. Change the root of $T_1 = (V_1, E_1)$ from r to r_t , where $t = \deg_T(r)$.
 5. For each sink $u \in S - r$, $\deg_T(u) > 0$, do:
 - $v \leftarrow parent_{T_1}(u)$; $k \leftarrow \deg_T(u)$
 - $E_1 \leftarrow E_1 - (u, v) + (u_k, v)$; $cost(u_k, v) \leftarrow cost(u, v)$.
 6. Output $T_1 = (V_1, E_1, \pi, cost)$.

FIG. 2. *The zero-skew stretching algorithm for arbitrary metric spaces.*

the cost of this path is

$$\begin{aligned}
 & cost(w, u_i) + cost(u_i, u_{i+1}) + \dots + cost(u_{k-1}, u_k) \\
 &= d(v_i, u) + \sum_{j=i}^{k-1} \{ [d(u, v_{j+1}) + delay_T(v_{j+1})] - [d(u, v_j) + delay_T(v_j)] \} \\
 &= [d(u, v_k) + delay_T(v_k)] - delay_T(v_i) \\
 &= delay_T(u) - delay_T(v_i).
 \end{aligned}$$

A similar computation shows that the cost of the path from u_k to u is $d(u, v_k) + delay_T(v_k) = delay_T(u)$.

The cost of T_1 is equal to $length(T)$ after Step 2 of the algorithm. In Step 3 it increases for each sink $u \in S$ by the cost of the path $(u, u_1, u_2, \dots, u_k)$, i.e., by $delay_T(u)$. Hence, the total cost of T_1 is

$$length(T) + \sum_{u \in S} delay_T(u) = length(T) + delay(T) \quad \square$$

3.2. Zero-skew stretching in metrically convex metric spaces. Before stating the algorithm, we need to introduce some more notation. A path $P = (p_1, p_2, \dots, p_k)$ in T_1 is called *critical* if it ends at a leaf node p_k and contains no loops. By construction, it follows that the tree T_1 produced by the algorithm in Figure 2 has at least one critical path starting from each node. Let $P = (p_1, p_2, \dots, p_k)$

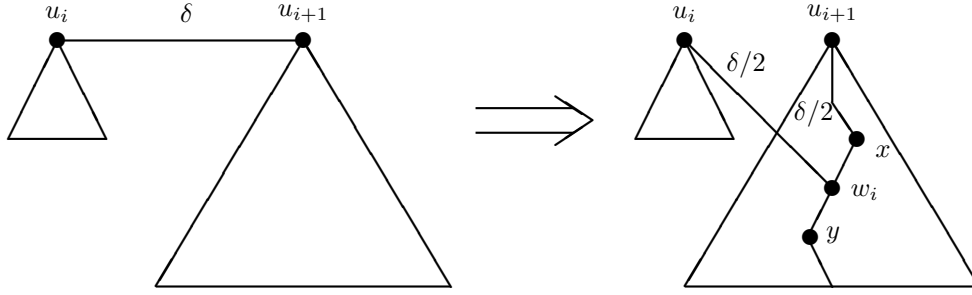


FIG. 3. Loop folding in metrically convex metric spaces.

Input: Rooted spanning tree $T = (S, E)$ in a metric space (M, d) .
Output: ZST $T_2 = (V_2, E_2, \pi, cost)$ for S .

1. Find $T_1 = (V_1, E_1, \pi, cost)$ using the algorithm in Figure 2.
2. $(V_2, E_2, \pi, cost) \leftarrow (V_1, E_1, \pi, cost)$.
3. For each sink $u \in S$ and $i = 0, 1, \dots, \deg_T(u)$, do:
 - // Add attachment node w_i on the critical path from u_{i+1}
 - Find edge $(x, y) = e(P, \delta/2)$ on the critical path P from u_{i+1} , where
 $\delta = cost(u_i, u_{i+1})$
 - $V_2 \leftarrow V_2 + w_i$; $\pi(w_i) \leftarrow v(P, \delta/2)$
 - $E_2 \leftarrow E_2 - (x, y) + (x, w_i) + (w_i, y)$
 - $cost(x, w_i) \leftarrow d(\pi(x), \pi(w_i))$; $cost(w_i, y) \leftarrow d(\pi(w_i), \pi(y))$
 - // Replace the loop (u_i, u_{i+1}) , where $u_0 \equiv u$, with the edge (u_i, w_i)
 - $E_2 \leftarrow E_2 - (u_i, u_{i+1}) + (u_i, w_i)$; $cost(u_i, w_i) \leftarrow \delta/2$.
4. Output $T_2 = (V_2, E_2, \pi, cost)$.

FIG. 4. The zero-skew stretching algorithm for metrically convex metric spaces.

be a critical path in T_1 , and let $length(P) = length(\pi(p_1), \pi(p_2), \dots, \pi(p_k))$. For every $0 \leq \delta \leq length(P)$, there exist i such that $length(\pi(p_1), \pi(p_2), \dots, \pi(p_i)) \leq \delta < length(\pi(p_1), \pi(p_2), \dots, \pi(p_{i+1}))$. We denote the edge (p_i, p_{i+1}) by $e(P, \delta)$. Since (M, d) is metrically convex, there is a point $v(P, \delta) \in M$ such that the $length(\pi(p_1), \dots, \pi(p_i), v(P, \delta)) = \delta$ and $length(v(P, \delta), \pi(p_{i+1}), \dots, \pi(p_k)) = length(P) - \delta$.

The improved stretching algorithm for metrically convex metric spaces (Figure 4) first computes a ZST T_1 using the algorithm in Figure 2. Then it “folds” half of each loop along a critical path of T_1 (see Figure 3). Folding can be applied to each loop (u_i, u_{i+1}) , since $cost(u_i, u_{i+1})$ is at most the length of the critical path P from u_{i+1} . Indeed, by Lemma 3.1, every path from u_{i+1} to a descendant leaf has the same cost. Hence, $cost(u_i, u_{i+1}) \leq cost(P)$. Finally, since P does not contain loops, each edge of P has cost equal to the distance between the embedding of its ends, and thus $cost(P) = length(P)$.

LEMMA 3.2. *The stretched tree T_2 produced by the algorithm in Figure 4 has zero-skew and total cost equal to $length(T) + delay(T)/2$.*

Proof. The total cost of the loops in the stretched tree T_1 is equal to $delay(T)$. Step 3 of the algorithm replaces each loop by an edge with half its cost. Therefore, $cost(T_2) = length(T) + delay(T)/2$. The tree T_2 has zero-skew, since T_1 has zero-skew,

Input: Finite set $S \subseteq M$.
Output: Rooted spanning tree T on S .

1. Initialization:
 - $ROOTS \leftarrow S; E \leftarrow \emptyset$
 - For each $v \in S$, $h(v) \leftarrow 0$.
2. While $|ROOTS| > 1$ do:
 - Find the closest two sinks $r, r' \in ROOTS$ with respect to metric d
 - If $h(r) < h(r')$, then swap r and r'
 - $E \leftarrow E + (r, r')$
 - $h(r) \leftarrow \max\{h(r), d(r, r') + h(r')\}$
 - $ROOTS \leftarrow ROOTS - r'$.
3. Output the tree $T = (S, E)$, rooted at the only remaining sink in $ROOTS$.

FIG. 5. *The Rooted-Kruskal algorithm.*

and loop folding preserves the cost of all root-to-leaf paths. \square

4. ZST approximation via spanning trees. In the previous section we have shown that any rooted spanning tree can be stretched into a ZST whose cost is equal to the length of the spanning tree plus its delay (half the delay, for metrically convex metric spaces). This motivates the following:

Zero-skew spanning tree problem. Given a set of points S in a (metrically convex) metric space (M, d) , find a rooted spanning tree T on S such that $cost(T) = length(T) + delay(T)$ (respectively, $length(T) + delay(T)/2$) is minimized.

Note that the MST on S has the shortest possible length but may have very large delay—if the MST is a simple path, then its delay may be as much as $O(n)$ times larger than its length. On the other hand, a star having the least delay may be $O(n)$ times longer than the MST.

In this section we give an algorithm for finding a rooted spanning tree which has both delay and length at most two times the minimum ZST cost. Therefore, our algorithm gives factor 4 and 3 approximations for the ZST problem in general and metrically convex metric spaces, respectively. Simultaneously, our algorithm gives factor 4 and 3 approximations for the zero-skew spanning tree problem in the respective metric spaces, since $cost(T)$ cannot be smaller than the cost of the minimum ZST.

The algorithm (Figure 5) can be thought of as a rooted version of the well-known Kruskal MST algorithm. At all times, the algorithm maintains a collection of rooted trees spanning the sinks; initially, each sink is a tree by itself. In each step, the algorithm chooses two trees that have the smallest distance between their roots and merges them by linking the root of one tree as the child of the other. In order to keep the delay of the resulting tree small, the child root is always chosen to be the root with smaller delay.

LEMMA 4.1. $delay(T) \leq length(T)$.

Proof. Note that, at the end of the Rooted-Kruskal algorithm, $h(u)$ represents exactly the delay of node u in T . Every iteration of the algorithm adds the edge (r, r') to $E(T)$, thus increasing $length(T)$ by $d(r, r')$. On the other hand, since $h(r) \geq h(r')$ when $h(r)$ is updated, the iteration contributes at most $d(r, r') + h(r') - h(r) \leq d(r, r')$ to $\sum_{u \in S} h(u)$, i.e., to the total delay of T . \square

Let n be the number of sinks in S .

LEMMA 4.2. $length(T) \leq 2(1 - 1/n)ZST^*(S)$.

Proof. Let s_1 be the root of T , and let s_2, \dots, s_n be the remaining $n - 1$ nodes of T , indexed in reverse order of their deletion from $ROOTS$. Since in each iteration the algorithm adds to T the edge joining a closest pair of points in $ROOTS$,

$$length(T) = \sum_{i=1}^{n-1} \text{MinDist}\{s_1, \dots, s_{i+1}\}.$$

Thus, by Lemma 2.1,

$$length(T) \leq 2 ZST^*(S) - \text{MinDist}\{s_1, s_2\} = 2 ZST^*(S) - d(s_1, s_2).$$

Since (s_1, s_2) is the longest edge in T , $d(s_1, s_2) \geq length(T)/(n - 1)$, and the lemma follows. \square

Lemmas 3.1, 4.1, and 4.2 give the following theorem.

THEOREM 4.3. *For any metric space and any set of n sinks, running the algorithm in Figure 2 on the tree T produced by the Rooted-Kruskal algorithm gives a ZST whose cost is at most $4(1 - 1/n)$ times larger than $ZST^*(S)$.*

Proof. By Lemma 3.1, the cost of the embedding is equal to $length(T) + delay(T)$. However, $delay(T) \leq length(T)$ by Lemma 4.1, and the approximation factor follows from Lemma 4.2. \square

Similarly, Lemmas 3.2, 4.1, and 4.2 give the following theorem.

THEOREM 4.4. *For any metrically convex metric space and any set of n sinks, running the algorithm in Figure 4 on the tree T produced by the Rooted-Kruskal algorithm gives a ZST whose cost is at most $3(1 - 1/n)$ times larger than $ZST^*(S)$.*

Proof. By Lemma 3.2, the cost of the embedding is now equal to $length(T) + (1/2) \cdot delay(T)$, and the theorem follows again from Lemmas 4.1 and 4.2. \square

The following example shows that the algorithm in Theorem 4.3 can produce ZSTs which are $4(1 - 1/n)$ times larger than optimal. A similar example shows that the algorithm in Theorem 4.4 has a tight approximation factor of $3(1 - 1/n)$.

Example 1. Consider a discrete metric space on $2^k + 1$ points, $n = 2^k$ of which are sinks. We label the sinks with 0-1 sequences of length k , i.e., $S = \{\alpha = b_{k-1}b_{k-2} \dots b_0 \mid b_i \in \{0, 1\}\}$. All sink-to-sink distances are equal to 1 and the distance from the single Steiner point to each of the sinks is $1/2$. In this space, the optimal ZST is a star rooted at the Steiner point and has cost equal to $n/2$. The Rooted-Kruskal algorithm may construct the spanning tree T with root $(11 \dots 1)$ and edges (α, α') such that α' is identical to α except that the rightmost 0 in α' is replaced with 1 in α . Indeed, at each iteration of Step 2, the algorithm may choose to merge trees rooted at α and α' as above. It may choose α to be the root of the merged tree, since $h(\alpha) = h(\alpha')$.

Clearly, $length(T) = n - 1$. On the other hand, since we always merge two roots with the same h -value, each merge contributes exactly 1 to the total delay of T . Thus, $delay(T) = n - 1$. By Lemma 3.1, the cost of the ZST produced by the algorithm is

$$length(T) + delay(T) = 2(n - 1) = 4(1 - 1/n) \cdot \frac{n}{2}.$$

Running time. The running time of the stretching algorithms given in section 3 is dominated by the time needed to sort the children of each node; this can be done in $O(n \log n)$ overall. For arbitrary metrics the Rooted-Kruskal algorithm can

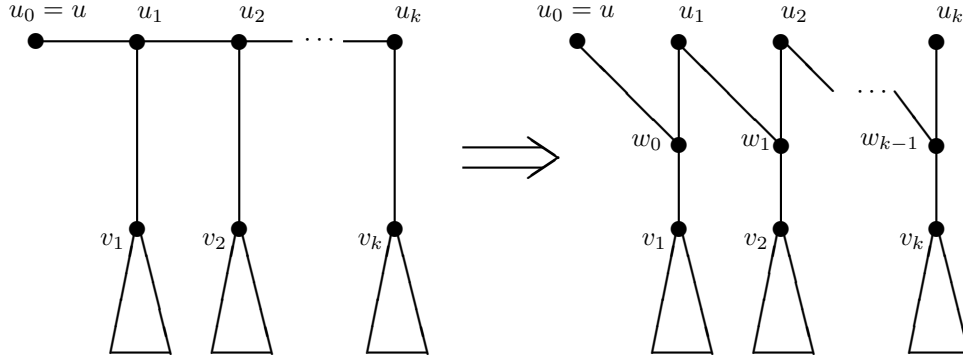


FIG. 6. When Figure 4 is applied to the Rooted-Kruskal spanning tree, the topology of the stretched tree remains the same, since each attachment node w_i belongs to the edge (u_{i+1}, v_{i+1}) .

be implemented in $O(n^2)$ time using Eppstein's dynamic closest-pair data structure [12]. In the rectilinear plane (in fact, in any fixed dimensional L_p space), the running time can be reduced to $O(n \log n)$ time by using the dynamic closest-pair data structure of Bespamyatnikh [4]. These implementations of the Rooted-Kruskal algorithm are asymptotically optimal, since the running times match known lower bounds for computing the first closest pair.

Thus, the total time for running the Rooted-Kruskal algorithm followed by one of the stretching algorithms given in section 3 is $O(n^2)$ in arbitrary metric spaces (respectively, $O(n \log n)$ in the rectilinear plane). Notice that this matches asymptotically the time needed for computing an MST for the sinks.

5. Practical considerations for approximating the rectilinear ZST.

In the previous two sections it has been shown that the minimum cost ZST can be approximated in metrically convex metric spaces within a factor of 3. In order to obtain better ZSTs in the rectilinear plane, we may combine the stretched spanning tree with the DME algorithm [5, 6, 10]. The DME algorithm gives the optimal rectilinear ZST for any given *topology*, which is an unweighted binary tree with the leaves labeled by the sinks. Therefore, we may shorten only the rectilinear ZST if we feed the topology of the stretched spanning tree into the DME algorithm.

In section 3 we suggested two different ways of stretching a spanning tree. One may expect that the topology produced by the algorithm in Figure 4 (the loop folding algorithm) is superior to the topology produced by the algorithm in Figure 2. Surprisingly, when stretching the spanning tree produced by the Rooted-Kruskal algorithm, both algorithms lead to the same topology. As proven below, every attachment node w_i inserted by the algorithm in Figure 4 belongs to the edge (u_{i+1}, v_{i+1}) . Hence, loop folding does not change the topology of the stretched tree (see Figure 6).

THEOREM 5.1. *Let T be the rooted spanning tree constructed by the Rooted-Kruskal algorithm. In any metrically convex metric space, the topologies produced by running Figures 2 and 4 on T are identical.*

Proof. Let the children $\{v_1, \dots, v_k\}$ of a node u be sorted as in the algorithm in Figure 2, i.e., in nondecreasing order of $d(u, v_i) + \text{delay}_T(v_i)$. For brevity, denote $d_i = d(u, v_i)$ and $D_i = \text{delay}_T(v_i)$. We will show that $\delta = \text{cost}(u_i, u_{i+1})$ is no greater than d_{i+1} . This will ensure that the attachment node w_i lies on the edge (u_{i+1}, v_{i+1}) and, therefore, the tree topologies produced by the two stretching algorithms are the

same (see Figure 6). Since $\delta = (d_{i+1} + D_{i+1}) - (d_i + D_i)$, it suffices to prove that

$$(5.1) \quad D_{i+1} \leq d_i + D_i.$$

We say that index k *precedes* index l if the node v_k has been attached to u before v_l in the Rooted-Kruskal algorithm. Let p_1 be the maximum index preceding $i + 1$, p_2 be the maximum index preceding p_1 , and so on, until we arrive at an index p_m with $D_{p_m} = 0$.⁴ Then $d_{p_1} + D_{p_1}$ represents the length of the critical path from u at the time when v_{i+1} is linked to u by the Rooted-Kruskal algorithm, and $d_{p_{i+1}} + D_{p_{i+1}}$ is the length of the critical path from u at the time when v_{p_i} is linked to u .

Notice that, since the distance between the closest two sinks in *ROOTS* does not decrease during the Rooted-Kruskal algorithm,

$$(5.2) \quad d_{i+1} \geq d_{p_1} \geq \dots \geq d_{p_m}.$$

Moreover,

$$(5.3) \quad D_{i+1} \leq d_{p_1} + D_{p_1}$$

and

$$(5.4) \quad D_{p_{j-1}} \leq d_{p_j} + D_{p_j}$$

for every $j = 2, \dots, m - 1$, since through all attachments node u remains the root.

Assume, for a contradiction, that (5.1) does not hold. We will show by induction on j that $p_j > i + 1$ and $D_{i+1} \leq D_{p_j}$ for every $j = 1, \dots, m$. Since $D_{p_m} = 0$, the above claim implies that $D_{i+1} = 0$, making (5.1) trivially true.

To prove the claim, first consider $j = 1$. If $p_1 \leq i$, then $d_{p_1} + D_{p_1} \leq d_i + D_i$, and (5.3) implies (5.1). So, it must be the case that $i + 1 < p_1$. Then $d_{i+1} + D_{i+1} \leq d_{p_1} + D_{p_1}$, and (5.2) implies that $D_{i+1} \leq D_{p_1}$.

Now assume that $D_{i+1} \leq D_{p_{j-1}}$ for some $j \geq 2$. If $p_j \leq i$, using (5.4) we get

$$D_{i+1} \leq D_{p_{j-1}} \leq d_{p_j} + D_{p_j} \leq d_i + D_i.$$

So, it must be the case that $i + 1 < p_j$. Then $d_{i+1} + D_{i+1} \leq d_{p_j} + D_{p_j}$, and, since $d_{i+1} \geq d_{p_j}$ by (5.2), this implies that $D_{i+1} \leq D_{p_j}$. \square

COROLLARY 5.2. *Combination of the Rooted-Kruskal algorithm with the stretching algorithm for arbitrary metric spaces (Figure 2) and with the DME algorithm gives a 3-approximation for the rectilinear ZST problem.*

6. Approximate BSTs. In this section we give two approximation algorithms for the BST problem, both built around a black-box ZST approximation algorithm. In both cases we construct a ZST for an appropriately chosen subset of the sinks, then extend this ZST to a b -BST for all sinks. In the first algorithm (Figure 7) the extension is done by adding subtrees of an MST on the sinks; in the second (Figure 8) subtrees are extracted from an approximate Steiner tree.

⁴We will always arrive at an index p_m with $D_{p_m} = 0$, since at least one child of u has zero delay. Indeed, let v be the child first connected to u . At the moment when the edge (u, v) is added by the Rooted-Kruskal algorithm, u has zero delay and thus v must also have zero delay. The delay of v never changes after its removal from *ROOTS*.

Input: Finite set $S \subseteq M$, bound $b > 0$.
Output: b -BST for S .

1. Find an MST T_0 on S , with respect to the metric d , and choose an arbitrary sink r as root.
2. Find a set W of sinks and a collection of subtrees of T_0 , $(B_u)_{u \in W}$, as follows:
 $W \leftarrow \emptyset; T \leftarrow T_0$
While $T \neq \emptyset$ do:
Find a sink v of T which is furthest from the root
Find the highest ancestor, say u , of v that still has $\text{delay}_T(u) \leq b$
 $W \leftarrow W + u; B_u \leftarrow T_u; T \leftarrow T - (u, \text{parent}(u)) - B_u$.
3. Find an approximate ZST, T_1 , for W .
4. Output the tree $T' = T_1 \cup (\bigcup_{u \in W} B_u)$ rooted at the root of T_1 .

FIG. 7. The MST based BST algorithm.

6.1. The MST based algorithm. The first algorithm (Figure 7) uses a simple iterative construction to cover the sinks by disjoint b -skew subtrees of an MST T_0 of S . The algorithm then outputs the union of these subtrees with a ZST T_1 on their roots. Clearly, the resulting tree T' is a b -BST for S . Moreover, $\text{cost}(T') \leq \text{cost}(T_1) + \text{length}(T_0)$, since the subtrees are disjoint pieces of T_0 . Hence, if the ZST algorithm used in Step 3 has an approximation factor of r_{ZST} , by Lemma 2.2 we get that

$$\begin{aligned} \text{cost}(T') &\leq r_{ZST} \text{ZST}^*(W) + \text{length}(T_0) \\ &\leq r_{ZST} (\text{BST}^*(S) + b \cdot (|W| - 1)) + \text{length}(T_0). \end{aligned}$$

For each node $u \neq r$ added to W in Step 2 of the algorithm in Figure 7, the path from the parent of u to the sink v is deleted from the tree. Since v is a furthest sink, the length of this path is equal to $\text{delay}_T(\text{parent}(u))$. By the choice of u , $\text{delay}_T(\text{parent}(u)) > b$. Thus, $b \cdot (|W| - 1) \leq \text{length}(T_0)$, and so

$$\text{cost}(T') \leq r_{ZST} \text{BST}^*(S) + (r_{ZST} + 1) \text{length}(T_0).$$

Let r_{MST} be the *Steiner ratio* for the metric space (M, d) , i.e., the supremum, over all sets of points S in (M, d) , of the ratio between the length of an MST and the length of a minimum Steiner tree for S . Since the length of the minimum Steiner tree for S is a lower bound on $\text{BST}^*(S)$, we get that $\text{length}(T_0) \leq r_{MST} \text{BST}^*(S)$. Hence, we have the following theorem.

THEOREM 6.1. *The algorithm in Figure 7 has an approximation factor of $r_{ZST} + r_{MST} + r_{ZST} r_{MST}$.*

Since the Steiner ratio is at most 2 for any metric space [18], and $3/2$ for the rectilinear plane [13], by using the results in Theorems 4.3 and 4.4 we get the following corollary.

COROLLARY 6.2. *The approximation factor of the algorithm in Figure 7 is 14 in arbitrary metric spaces, 11 in arbitrary metrically convex metric spaces, and 9 in the rectilinear plane.*

Notice that the running time of the algorithm in Figure 7 is still $O(n \log n)$ for the rectilinear plane and $O(n^2)$ for arbitrary metric spaces. The MST in Step 1 can

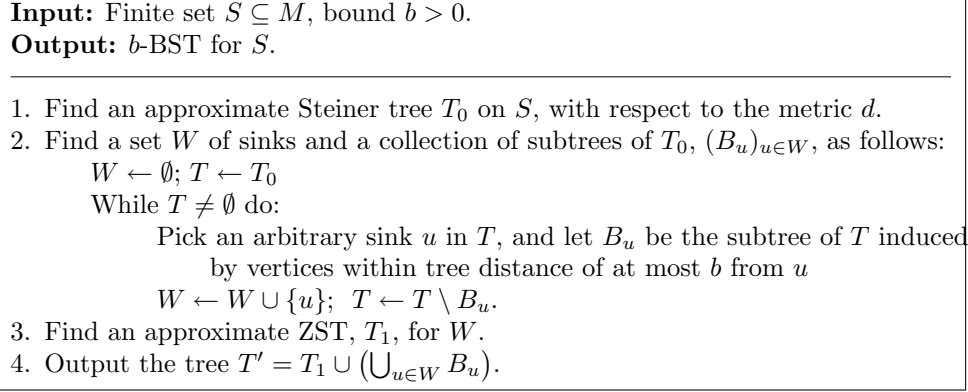


FIG. 8. The approximate Steiner tree based BST algorithm.

be computed within these time bounds using Hwang's [14] rectilinear MST algorithm and Kruskal's algorithm, respectively, while Step 2 can be implemented in linear time.

6.2. The approximate Steiner tree based algorithm. The second BST algorithm combines a ZST for a subset W of the sinks with b -skew subtrees of an approximate Steiner tree T_0 (Figure 8).

THEOREM 6.3. *The BST problem can be approximated within a factor of $r_{ZST} + r_{SMT} + 2r_{ZST}r_{SMT}$, given r_{ZST} (respectively, r_{SMT}), approximation algorithms for the ZST, and minimum Steiner tree problems.*

Proof. By construction, the distance in T_0 between any two sinks in W is at least b . Consider the set of open balls of radius $b/2$ centered at the sinks in W , with the balls considered in the metric space induced by T_0 . Since any two such balls are disjoint, and each of them must cover at least $b/2$ worth of edges of T_0 , we get that

$$(6.1) \quad b|W| \leq 2 \text{length}(T_0).$$

To estimate the cost of the BST produced by the algorithm, notice that $\bigcup_{u \in W} B_u$ has total cost of at most $\text{length}(T_0)$. By Lemma 2.2 and (6.1), we get

$$\begin{aligned} \text{cost}(T') &\leq r_{ZST} \text{ZST}^*(W) + \text{length}(T_0) \\ &\leq r_{ZST} (\text{BST}^*(S) + b \cdot (|W| - 1)) + \text{length}(T_0) \\ &\leq r_{ZST} (\text{BST}^*(S) + 2 \text{length}(T_0)) + \text{length}(T_0), \end{aligned}$$

and the theorem follows by observing that $\text{length}(T_0) \leq r_{SMT} \text{BST}^*(S)$, since, as noted above, the length of the minimum Steiner tree for S is a lower bound on $\text{BST}^*(S)$. \square

With the currently known approximation factors for Steiner trees and ZST, Theorem 6.1 gives better BST approximations than Theorem 6.3 for the rectilinear plane, as well as arbitrary (metrically convex) metric spaces. However, Theorem 6.3 may improve upon Theorem 6.1 for metric spaces with good Steiner tree approximation (r_{SMT} close to 1) and large Steiner ratio (r_{MST} close to 2), e.g., for high dimensional L_p spaces.

7. Conclusions and open problems. We have given approximation algorithms for the ZST and BST problems with improved approximation factors for

general and metrically convex metric spaces, as well as the rectilinear plane. Our algorithms have a practical running time: $O(n \log n)$ in the rectilinear plane and $O(n^2)$ in general metric spaces. Preliminary experiments also show that, when combined with the linear time DME algorithm of [5, 6, 10], our rectilinear ZST algorithm gives results competitive to those obtained by the Greedy DME heuristic of Edahiro [11], which is regarded in the VLSI CAD community as the best ZST heuristic to date (see [17]).

An interesting open question is to determine the limitations of the spanning-tree based ZST construction introduced in this paper. One can define the *zero-skew Steiner ratio* of a metric space as the supremum, over all sets of sinks, of the ratio between the minimum zero-skew cost (i.e., *length + delay*) of a spanning tree and the minimum ZST cost. The results in section 4 imply that the zero-skew Steiner ratio is at most 4 in arbitrary metric spaces, and at most 3 in metrically convex metric spaces. On the other hand, we have constructed instances showing that the zero-skew Steiner ratio can be as large as 3 for arbitrary metric spaces; we conjecture that the ratio is never larger than 3. Determining the complexity of the zero-skew spanning tree problem is another interesting open question.

In the *planar* versions of the rectilinear ZST and BST problems, one seeks zero and bounded-skew trees in the rectilinear plane with no self-intersecting edges. Charikar et al. [8] have given the first constant approximation factors for these versions; it would be interesting to find algorithms with improved approximation factors.

REFERENCES

- [1] S. ARORA, *Polynomial time approximation schemes for Euclidean TSP and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.
- [2] H. BAKOGLU, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.
- [3] H. BAKOGLU, J. WALKER, AND J. MEINDL, *A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock-skew in ULSI and WSI circuits*, in Proceedings of the IEEE International Conference on Computer Design, 1986, pp. 118–122.
- [4] S. BESPAMYATNIK, *An optimal algorithm for closest-pair maintenance*, Discrete Comput. Geom., 19 (1998), pp. 175–195.
- [5] K. BOESE AND A. KAHNG, *Zero-skew clock routing trees with minimum wirelength*, in Proceedings of the IEEE International ASIC Conference, 1992, pp. 17–21.
- [6] T. H. CHAO, Y.-C. HSU, AND J.-M. HO, *Zero skew clock net routing*, in Proceedings of the ACM/IEEE Design Automation Conference, 1992, pp. 518–523.
- [7] T.-H. CHAO, Y.-C. HSU, J.-M. HO, K. BOESE, AND A. KAHNG, *Zero skew clock routing with minimum wirelength*, IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process., 39 (1992), pp. 799–814.
- [8] M. CHARIKAR, J. KLEINBERG, R. KUMAR, S. RAJAGOPALAN, A. SAHAI, AND A. TOMKINS, *Minimizing wirelength in zero and bounded skew clock trees*, in Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, ACM, New York, 1999, pp. 177–184.
- [9] J. CONG, A. KAHNG, C. KOH, AND C.-W. TSAO, *Bounded-skew clock and Steiner routing*, ACM Trans. on Design Automation of Electronic Systems, 3 (1998), pp. 341–388.
- [10] M. EDAHIRO, *Minimum skew and minimum path length routing in VLSI layout design*, NEC Res. Development, 32 (1991), pp. 569–575.
- [11] M. EDAHIRO, *A clustering-based optimization algorithm in zero-skew routings*, in Proceedings of the 30th ACM/IEEE Design Automation Conference, 1993, pp. 612–616.
- [12] D. EPPSTEIN, *Fast hierarchical clustering and other applications of dynamic closest pairs*, ACM J. Exp. Algorithmics, 5 (2000), pp. 1–23.
- [13] F. K. HWANG, *On Steiner minimal trees with rectilinear distance*, SIAM J. Appl. Math., 30 (1976), pp. 104–114.
- [14] F. K. HWANG, *An $O(n \log n)$ algorithm for rectilinear minimal spanning trees*, J. ACM, 26 (1979), pp. 177–182.

- [15] M. JACKSON, A. SRINIVASAN, AND E. KUH, *Clock routing for high-performance ICs*, in Proceedings of the ACM/IEEE Design Automation Conference, 1990, pp. 574–579.
- [16] A. B. KAHNG, J. CONG, AND G. ROBINS, *High-performance clock routing based on recursive geometric matching*, in Proceedings of the ACM/IEEE Design Automation Conference, 1990, pp. 574–579.
- [17] A. B. KAHNG AND G. ROBINS, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, Norwell, MA, 1995.
- [18] L. KOU, G. MARKOWSKY, AND L. BERMAN, *A fast algorithm for Steiner trees*, Acta Inform., 15 (1981), pp. 141–145.
- [19] Y. LI AND M. JABRI, *A zero-skew clock routing scheme for VLSI circuits*, in Proceedings of the IEEE International Conference on Computer-Aided Design, 1992, pp. 458–463.
- [20] G. N. ROUSKAS AND I. BALDINE, *Multicast routing with end-to-end delay and delay variation constraints*, IEEE J. on Selected Areas in Communications, 15 (1997), pp. 346–356.
- [21] G. ROBINS AND A. ZELIKOVSKY, *Improved Steiner tree approximation in graphs*, in Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 2000, ACM, New York, 2000, pp. 770–779.
- [22] A. ZELIKOVSKY AND I. MĂNDOIU, *Practical approximation algorithms for zero- and bounded-skew trees*, in Proceedings of the Twelfth ACM-SIAM Annual Symposium on Discrete Algorithms, 2001, ACM, New York, 2001, pp. 407–416.

FACETS OF THE WEAK ORDER POLYTOPE DERIVED FROM THE INDUCED PARTITION PROJECTION*

JEAN-PAUL DOIGNON[†] AND SAMUEL FIORINI[†]

Abstract. The weak order polytopes are studied in Gurgel and Wakabayashi [*Discrete Math.*, 175 (1997), pp. 163–172], Gurgel and Wakabayashi [*The Complete Pre-Order Polytope: Facets and Separation Problem*, manuscript, 1996], and Fiorini and Fishburn [*Weak order polytopes*, submitted]. We make use of their natural, affine projection onto the partition polytopes to determine several new families of facets for them. It turns out that not all facets of partition polytopes are lifted into facets of weak order polytopes. We settle the cases of all facet-defining inequalities established for partition polytopes by Grötschel and Wakabayashi [*Math. Programming*, 47 (1990), pp. 367–387]. Our method, although rather simple, allows us to establish general families of facets which contain two particular cases previously requiring long proofs.

Key words. weak order polytope, partition polytope, facet of convex polytope

AMS subject classifications. 52B12, 06A07, 90C57

PII. S0895480100369936

1. Introduction. In order to solve real-life problems which require finding an optimal linear ordering, Grötschel, Jünger, and Reinelt [8] have studied the facial structure of the “linear ordering polytope.” They found several facets and used these results in the so-called branch-and-cut technique which combines branch-and-bound with cutting planes techniques. The same polytope had appeared before under the name “binary choice polytope” in the theory of probabilistic utility; see the references in Fishburn [7]. Recently, several papers were devoted to the geometry of this polytope and closely related ones; see Fiorini and Fishburn [6] or Fiorini [5] and their bibliographies.

For a general definition, let \mathcal{F} be a family of reflexive relations on the set $n = \{1, 2, \dots, n\}$. Each element R of \mathcal{F} is encoded by its characteristic vector x^R , which has a coordinate $x_{(i,j)}^R$ for each pair (i, j) of distinct elements in n ; this coordinate equals 1 when iRj , and 0 otherwise. A subset of vertices of the unit cube in $\mathbb{R}^{n(n-1)}$ is thus associated to \mathcal{F} . The study of its convex hull, called the \mathcal{F} -polytope, includes, for instance, the determination of (many) facets, or, more ambitiously, of the full combinatorial structure. Several particular cases have been investigated. In the previous paragraph, \mathcal{F} is the family of all linear orderings on n . (For recent references, see, e.g., [3] and [4].) When \mathcal{F} is the set of all equivalence relations on n , the \mathcal{F} -polytope is also called the *partition polytope* P_{PA}^n , or the *clique-partitioning polytope*; see Grötschel and Wakabayashi [9]. Here we focus on the *weak order polytope* P_{WO}^n . A *weak order* on n is a reflexive, transitive, and complete relation on n . Such a weak order W is a ranking of the elements of n with ties allowed, typically

$$(1.1) \quad C_1 \prec C_2 \prec \dots \prec C_k,$$

where $\{C_1, C_2, \dots, C_k\}$ is the partition of n corresponding to the equivalence relation $W \cap W^{-1}$. The mapping $W \mapsto W \cap W^{-1}$ leads to the “induced partition projection”

*Received by the editors March 17, 2000; accepted for publication (in revised form) December 7, 2001; published electronically January 16, 2002.

<http://www.siam.org/journals/sidma/15-1/36993.html>

[†]Université Libre de Bruxelles, c.p. 216, Bd du Triomphe, B-1050 Brussels, Belgium (doignon@ulb.ac.be, sfiorini@ulb.ac.be).

from the weak order polytope P_{WO}^n onto the partition polytope P_{PA}^n , which we will exploit to produce new facets of weak order polytopes.

Adjacency of vertices in the weak order polytope P_{WO}^n is studied by Gurgel and Wakabayashi [12], while facets are determined by Gurgel and Wakabayashi [11] (see also Gurgel [10]). The results of these two papers are extended by Fiorini and Fishburn [6], who also spell out some motivation for the study of P_{WO}^n . By “lifting” known facets from P_{PA}^n (that is, taking in P_{WO}^n preimages of facets of P_{PA}^n), we derive several new families of facets of P_{WO}^n . For instance, two isolated examples of facets are given by Gurgel and Wakabayashi [11], the 20-page proof for one of them being omitted (see, however, Gurgel [10]); they are the first two instances of a general family of facets which we establish easily along our approach.

It should be stressed that not any facet of P_{PA}^n has a preimage which is a facet of P_{WO}^n . Thus a case by case analysis is required, whose outcome is reported here for all facet-defining inequalities established for P_{PA}^n by Grötschel and Wakabayashi [9]. (Additional facets of P_{PA}^n were recently provided in [1]; they are reserved for later work.) In summary, we derive facets of P_{WO}^n from trivial, 2-chorded even wheel, and 2-chorded path facets of P_{PA}^n , and also from 2-partition facets, except for the “smallest” one. On the other hand, triangle and 2-chorded cycle facets are lifted into ridges; we describe the two facets of P_{WO}^n containing such a ridge. Thus we show how a natural relation between the two polytopes can be useful in their study.

2. Permutation subspaces. Assume $n \geq 2$ throughout the paper. The weak order polytope P_{WO}^n lies by definition in the real affine space $\mathbb{R}^{n(n-1)}$ and has one vertex x^W for each weak order W on $n = \{1, 2, \dots, n\}$. Its dimension equals $n(n-1)$. On the other hand, as equivalence relations are symmetric, it is simpler to see the partition polytope P_{PA}^n in the real affine space $\mathbb{R}^{\binom{n}{2}}$ in which any point has one coordinate $y_{\{i,j\}}$ for each unordered pair $\{i, j\}$ in n . In fact, P_{PA}^n is of dimension $\binom{n}{2}$. Its vertex corresponding to the equivalence relation R will be denoted as y^R .

The mapping $W \mapsto E = W \cap W^{-1}$, where W is a weak order on n and E is the equivalence relation of W , provides a mapping from $\text{vert } P_{\text{WO}}^n$ (the set of all vertices of the weak order polytope) onto $\text{vert } P_{\text{PA}}^n$: in the same notation, x^W is mapped onto y^E . The latter mapping extends to the affine projection (i.e., surjective, affine mapping)

$$(2.1) \quad \pi : \mathbb{R}^{n(n-1)} \rightarrow \mathbb{R}^{\binom{n}{2}} : x \mapsto y \quad \text{with} \quad y_{\{i,j\}} = x_{(i,j)} + x_{(j,i)} - 1,$$

called the *induced partition projection*. As $\pi(P_{\text{WO}}^n) = P_{\text{PA}}^n$, the linear mapping π induces an inclusion preserving correspondence from the face lattice of P_{PA}^n to the face lattice of P_{WO}^n . If $f(y) \leq b$ is a linear inequality valid for P_{PA}^n which defines the face F , then $\hat{\pi}^{-1}(F) = P_{\text{WO}}^n \cap \pi^{-1}(F)$ is the face of P_{WO}^n defined by the inequality $(f \circ \pi)(x) \leq b$ (which is valid for P_{WO}^n). We say that the latter inequality is *lifted* from inequality $f(y) \leq b$ and also that the preimage $\hat{\pi}^{-1}(F)$ is *lifted* from F ; here $\hat{\pi}$ denotes the restriction of π to P_{WO}^n .

In particular, when the face F consists of a single vertex y^E , its preimage $\hat{\pi}^{-1}(y^E)$ is a face of P_{WO}^n which is affinely equivalent to the linear ordering polytope P_{LO}^k , with k the number of equivalent classes of E ; indeed, vertices of $\hat{\pi}^{-1}(y^E)$ correspond to all weak orders which linearly order the k classes of E .

The (*affine*) *dimension* of a set S of points in $\mathbb{R}^{n(n-1)}$ is the dimension of the affine subspace $\text{aff } S$ it spans and is denoted as $\dim S$. Now choose any point o in the affine space $\mathbb{R}^{n(n-1)}$ as an origin, thus transforming $\mathbb{R}^{n(n-1)}$ into a vector space $\mathbb{R}_o^{n(n-1)}$. (This point does not need to be $(0, 0, \dots, 0)$.) The *rank* $\text{rk } S$ of a set S of

points in $\mathbb{R}_o^{n(n-1)}$ is the rank of the vector subspace it generates.

Given a vertex y^E of \mathbb{P}_{PA}^n , the *permutation subspace* $\text{permsub}(y^E)$ is the (unique) vector subspace of $\mathbb{R}_o^{n(n-1)}$ that forms a translate of $\text{aff } \dot{\pi}^{-1}(y^E)$. Equivalently, $\text{permsub}(y^E)$ consists of all linear combinations of vectors \vec{op} with $p \in \dot{\pi}^{-1}(y^E)$, whose coefficients sum up to 0. Although $\dim \pi^{-1}(y^E) = \binom{n}{2}$, the face $\dot{\pi}^{-1}(y^E)$ can have a lower affine dimension; the latter equals $\text{rk } \text{permsub}(y^E)$. A basis for $\text{permsub}(y^E)$ can be easily found, as we now indicate. The trick is to consider the difference between two vertices $x^W, x^{W'}$ of $\dot{\pi}^{-1}(y^E)$ with W and W' , disagreeing only in the transposition of two consecutive classes. (The argument is classic for the linear ordering polytope.)

For two nonempty, disjoint subsets U and V of \mathfrak{n} , define in $\mathbb{R}_o^{n(n-1)}$ the *transposition vector* $\text{transp}(U, V)$, located at o , by specifying as follows its components in the canonical base (also located at o):

$$(2.2) \quad (\text{transp}(U, V))_{(i,j)} = \begin{cases} 1 & \text{if } i \in U \text{ and } j \in V, \\ -1 & \text{if } i \in V \text{ and } j \in U, \\ 0 & \text{otherwise.} \end{cases}$$

The set of all transposition vectors $\text{transp}(U, V)$, for U, V distinct classes of the equivalence relation E , span $\text{permsub}(y^E)$ but is linearly dependent (since $\text{transp}(U, V) = -\text{transp}(V, U)$). A basis of $\text{permsub}(y^E)$ is formed by selecting one of the two transposition vectors $\text{transp}(U, V)$ and $\text{transp}(V, U)$ for any unordered pair $\{U, V\}$ of classes of E . Such vectors $\text{transp}(U, V)$, with U and V two classes of E , are also called *transposition vectors* of the vertex y^E . When $U = \{u\}$, $V = \{v\}$, we abbreviate the notation by writing $\text{transp}(u, v)$.

PROPOSITION 2.1. *Let E and E' be two equivalence relations on \mathfrak{n} with $E \subseteq E'$. Then $\text{permsub}(y^E) \supseteq \text{permsub}(y^{E'})$.*

Proof. From the assumption, any two classes U', V' of E' are unions of classes of E , say $U' = \cup_{i=1}^k U_i$ and $V' = \cup_{j=1}^l V_j$. Then

$$(2.3) \quad \text{transp}(U', V') = \sum_{i=1}^k \sum_{j=1}^l \text{transp}(U_i, V_j),$$

and $\text{transp}(U', V') \in \text{permsub}(y^E)$. \square

If F is a nonempty face of \mathbb{P}_{PA}^n , its *permutation subspace* $\text{permsub}(F)$ is the vector subspace spanned by the union of the permutation subspaces of the vertices of F . By Proposition 2.1, we can ignore here any vertex $y^{E'}$ of F for which there exists a vertex y^E of F with $E \subset E'$.

The next result is the main tool we will use to derive facets of \mathbb{P}_{WO}^n .

PROPOSITION 2.2. *For any face F of \mathbb{P}_{PA}^n ,*

$$(2.4) \quad \dim F + \text{rk } \text{permsub}(F) \leq \dim \dot{\pi}^{-1}(F) \leq \dim F + \binom{n}{2}.$$

Proof. Choose a vertex o' of F as an origin in $\mathbb{R}^{\binom{n}{2}}$ and then choose a vertex o of $\dot{\pi}^{-1}(F)$ as an origin in $\mathbb{R}^{n(n-1)}$. As $\pi(o) = o'$, the affine mapping π becomes a linear mapping. Setting $f = \text{rk } F$, we may select f linearly independent vertices q_1, q_2, \dots, q_f of F . Pick in $\dot{\pi}^{-1}(F)$ vertices p_1, p_2, \dots, p_f with $\pi(p_i) = q_i$ for $i = 1, 2, \dots, f$. With $g = \text{rk } \text{permsub}(F)$, select next a basis r_1, r_2, \dots, r_g of $\text{permsub}(F)$ which consists of transposition vectors of vertices of F . Thus any r_j is a difference between

two vertices of $\dot{\pi}^{-1}(F)$. As is easily checked, the vectors $p_1, p_2, \dots, p_f, r_1, r_2, \dots, r_g$ are linearly independent. Because with o they all belong to $\text{aff } \dot{\pi}^{-1}(F)$, we have $f + g \leq \dim \dot{\pi}^{-1}(F)$, which is the first inequality to be proved. The second inequality follows at once from $\text{rk } \pi^{-1}(o') = \binom{n}{2}$. \square

Remark. Both inequalities in Proposition 2.2 become equalities at least when y^{id} belongs to F (where id is the equivalence relation with n classes) but not always (as will be seen later, e.g., in Proposition 4.1).

As we now show, Proposition 2.2 greatly helps understanding preimages by $\dot{\pi}$ of various facets of P_{PA}^n .

3. Lifting the trivial, triangle, and 2-partition inequalities. For i, j distinct elements in n , inequality $x_{\{ij\}} \geq 0$ defines a facet of P_{PA}^n ; see Grötschel and Wakabayashi [9]. (Notice how we sometime abbreviate indices $\{i, j\}$ in $\{ij\}$.) The lifted inequality $x_{ij} + x_{ji} - 1 \geq 0$ defines a facet of P_{WO}^n as shown by Gurgel and Wakabayashi [12]. (In index position, (i, j) is from now on abbreviated in ij or i, j .) On the other hand, if F is the facet of P_{PA}^n defined by the *triangle inequality*

$$(3.1) \quad x_{\{ij\}} + x_{\{jk\}} - x_{\{ik\}} \leq 1,$$

where i, j, k are distinct elements in n , the preimage $\dot{\pi}^{-1}(F)$ defined by

$$(3.2) \quad x_{ij} + x_{ji} + x_{jk} + x_{kj} - x_{ik} - x_{ki} \leq 2$$

turns out to be a ridge. (We skip the proof.) This ridge is the intersection of the two facets defined by the two following *transitivity inequalities* [11], [10]:

$$\begin{aligned} x_{ij} + x_{jk} - x_{ik} &\leq 1, \\ x_{kj} + x_{ji} - x_{ki} &\leq 1. \end{aligned}$$

Now consider two nonempty, disjoint subsets S, T of n with $|S| < |T|$. Grötschel and Wakabayashi [9] show the *2-partition inequality*

$$(3.3) \quad \sum_{\substack{i \in S \\ j \in T}} x_{\{ij\}} - \sum_{\substack{i, j \in S \\ i < j}} x_{\{ij\}} - \sum_{\substack{i, j \in T \\ i < j}} x_{\{ij\}} \leq |S|$$

to be facet-defining for P_{PA}^n . Its lifted inequality in $\mathbb{R}^{n(n-1)}$, that we also call a *2-partition inequality*,

$$(3.4) \quad \sum_{\substack{i \in S \\ j \in T}} (x_{ij} + x_{ji}) - \sum_{\substack{i, j \in S \\ i \neq j}} x_{ij} - \sum_{\substack{i, j \in T \\ i \neq j}} x_{ij} \leq |S| + |S||T| - \binom{|S|}{2} - \binom{|T|}{2}$$

is valid for P_{WO}^n . When $S = \{j\}$ and $T = \{i, k\}$ this inequality coincides with inequality (3.2), so it is clearly not facet-defining.

THEOREM 3.1. *Let S and T be two disjoint subsets of n such that $0 < |S| < |T|$ and $(|S|, |T|) \neq (1, 2)$. Then inequality (3.4) defines a facet of P_{WO}^n .*

Proof. According to the trivial lifting lemma of Fiorini and Fishburn [6], any facet-defining inequality for P_{WO}^n is also facet-defining for P_{WO}^m when $m > n$. We may thus assume $S \cup T = n$. Let F be the facet of P_{PA}^n defined by inequality (3.3). Then, inequality (3.4) defines the preimage $\dot{\pi}^{-1}(F)$ in P_{WO}^n . We proceed by showing that all transposition vectors $\text{transp}(i, j)$, for i, j distinct in n , belong to $\text{permsub}(F)$; the thesis then follows from Proposition 2.2.

For every injective mapping $f : S \rightarrow T$, an equivalence relation E yields $y^E \in F$ when it has the classes $\{s, f(s)\}$ for all s in S and $\{t\}$ for all t in $T \setminus f(S)$.

When $|S| + 2 \leq |T|$, we deduce that all transposition vectors $\text{transp}(j, k)$, for $j, k \in T$, and $j \neq k$, belong to $\text{permsub}(F)$. Also, for $i \in S$ and $j, k \in T$ with $j \neq k$, we get $\text{transp}(\{j\}, \{i, k\}) \in \text{permsub}(F)$ which, after subtraction of $\text{transp}(j, k)$, gives $\text{transp}(j, i) \in \text{permsub}(F)$. If S contains two distinct elements h, i , similar arguments give $\text{transp}(h, i) \in \text{permsub}(F)$. Thus $\text{permsub}(F)$ contains all $\text{transp}(u, v)$ for u, v distinct in $S \cup T = \mathfrak{n}$.

When $|S| + 1 = |T|$, we have $|S| \geq 2$ by assumption. Let i, j be two distinct elements in S and k, l be two distinct elements in T . By considering appropriate partitions encoded as vertices of F , we find the following six linearly independent transposition vectors in $\text{permsub}(F)$:

$$\begin{aligned} & \text{transp}(k, i) + \text{transp}(k, l), \\ & \text{transp}(k, j) + \text{transp}(k, l), \\ & \text{transp}(l, i) + \text{transp}(l, k), \\ & \text{transp}(l, j) + \text{transp}(l, k), \\ & \text{transp}(k, j) + \text{transp}(k, l) + \text{transp}(i, j) + \text{transp}(i, l), \\ & \text{transp}(i, j) + \text{transp}(i, k) + \text{transp}(l, j) + \text{transp}(l, k). \end{aligned}$$

All transposition vectors $\text{transp}(u, v)$ with u, v distinct in $\{i, j, k, l\}$ are linear combinations of the six vectors in the above list. Thus again $\text{permsub}(F)$ contains all possible transposition vectors. \square

4. Lifting the 2-chorded cycle inequalities. Consider a cycle in \mathfrak{n} ; to simplify notation, we relabel the elements of \mathfrak{n} in such a way that the cycle has vertices $1, 2, \dots, k$. We denote by \oplus and \ominus the addition and subtraction on $\mathfrak{k} = \{1, 2, \dots, k\}$ with results reduced modulo k to a value in \mathfrak{k} . The *2-chorded cycle inequality*

$$(4.1) \quad \sum_{i=1}^k x_{\{i, i \oplus 1\}} - \sum_{i=1}^k x_{\{i, i \oplus 2\}} \leq \frac{k-1}{2}$$

is facet-defining for $P_{\text{PA}}^{\mathfrak{n}}$ when k is odd and at least 5 (see [9]). In what follows, we assume this double condition on k . Taking the preimage by the induced partition projection π , we derive the following inequality valid for $P_{\text{WO}}^{\mathfrak{n}}$:

$$(4.2) \quad \sum_{i=1}^k (x_{i, i \oplus 1} + x_{i \oplus 1, i}) - \sum_{i=1}^k (x_{i, i \oplus 2} + x_{i \oplus 2, i}) \leq \frac{k-1}{2}.$$

As will be shown in Proposition 4.1, this inequality defines a ridge of $P_{\text{WO}}^{\mathfrak{n}}$; the two facets containing the ridge are specified in Theorem 4.2. We use the notation

$$\begin{aligned} \mathcal{D}_k &= \{(i, j) \in \mathfrak{k} \times \mathfrak{k} \mid i \ominus j \text{ is odd and } i \neq j\}, \\ \mathcal{E}_k &= \{(i, j) \in \mathfrak{k} \times \mathfrak{k} \mid i \ominus j \text{ is even}\}. \end{aligned}$$

PROPOSITION 4.1. *For k odd and at least 5, inequality (4.2) defines a ridge of $P_{\text{WO}}^{\mathfrak{n}}$. Any vertex of $P_{\text{WO}}^{\mathfrak{n}}$ satisfying inequality (4.2) with equality also satisfies the linear equation*

$$(4.3) \quad \sum_{ij \in \mathcal{E}_k} x_{ij} - \sum_{ij \in \mathcal{D}_k} x_{ij} = 0.$$

Proof. Let F denote the facet of P_{PA}^n defined by inequality (4.1); then inequality (4.2) defines the face $G = \dot{\pi}^{-1}(F)$ of P_{WO}^n . We show that any vertex x^W of G also satisfies (4.3). Indeed, $\pi(x^W) \in F$, and a vertex y^E belongs to F iff the equivalence relation E , hereafter denoted as \sim , enjoys up to cyclic rotation of $1, 2, \dots, k$ either the conditions

$$(4.4) \quad 1 \not\sim 2 \sim 3 \not\sim 4 \sim 5 \not\sim 6 \sim 7 \not\sim \dots \not\sim k-1 \sim k \not\sim 1$$

or the conditions

$$(4.5) \quad 1 \sim 2 \sim 3 \not\sim 4 \sim 5 \not\sim 6 \sim 7 \not\sim \dots \not\sim k-1 \sim k \not\sim 1.$$

Either set of conditions implies that x^W also satisfies (4.3).

Now to prove that G is a ridge, it suffices by Proposition 2.2 to prove that $\text{permsub}(F)$ plus the transposition vector $\text{transp}(1, 2)$ generate all transposition vectors $\text{transp}(i, j)$ with i, j distinct in n . Take a minimal equivalence relation E such that y^E satisfies (4.1) with equality. Then E admits the class $\{i\}$ for all $i \in n \setminus k$; moreover, we may assume, after relabeling if necessary, that E admits the class $\{1\}$. As $y^E \in F$, we get $\text{transp}(1, i)$ and $\text{transp}(i, j)$ in $\text{permsub}(F)$ for all i, j distinct in $n \setminus k$. Taking appropriate cyclic images of E , we see that the same holds if 1 is replaced with any element in k . Now using the minimal equivalence relation defined by (4.4) together with its cyclically rotated images, we can check that $\text{permsub}(F)$ plus $\text{transp}(1, 2)$ generate $\text{transp}(1, 3)$, $\text{transp}(2, 3)$, $\text{transp}(2, 4)$, $\text{transp}(3, 4)$, $\text{transp}(1, 4)$, $\text{transp}(3, 5)$, etc., thus all $\text{transp}(i, j)$ with i, j distinct in k . \square

THEOREM 4.2. *Let k be odd and at least 5. One of the two facets of P_{WO}^n containing the ridge obtained in Proposition 4.1 is defined by the inequality*

$$(4.6) \quad \frac{k+1}{2} \sum_{i=1}^k (x_{i, i \oplus 1} - x_{i, i \oplus 2}) + \frac{k-3}{2} \sum_{i=1}^k (x_{i \oplus 1, i} - x_{i \oplus 2, i}) \\ + \sum_{j=3}^{(k-1)/2} \sum_{i=1}^k (-1)^{j+1} (x_{i, i \oplus j} - x_{i \oplus j, i}) \leq \frac{(k-1)^2}{4},$$

and the other facet containing the ridge is defined by the similar inequality obtained by substituting x_{ij} with x_{ji} for all i, j in k with $i < j$.

Gurgel and Wakabayashi [11] present two facet-defining inequalities for P_{WO}^n which are the two particular cases of Theorem 4.2 for $k = 5$ and $k = 7$; the 20-page proof for $k = 7$ is omitted there but appears in Gurgel [10]. Our proof of Theorem 4.2 is rather short and uses the following known result on tournaments (see Bermond [2] or Laslier [13]).

LEMMA 4.3. *For k odd, the relation \mathcal{D}_k is a tournament on k whose Slater index (or feedback arc-number) equals $\frac{k^2-1}{8}$. The same holds for the relation \mathcal{E}_k .*

Proof. As k is assumed to be odd, $i \ominus j$ is even iff $j \ominus i$ is odd; thus \mathcal{D}_k is a tournament. Let L be any linear ordering on k obtained by reversing certain arcs of \mathcal{D}_k . For $i = 1, 2, \dots, (k-1)/2$, the outdegree of the element of rank i in L equals $k-i$, while in \mathcal{D}_k it equals $(k-1)/2$. Thus the number of reversed arcs is at least $\sum_{i=1}^{(k-1)/2} ((k-i) - (k-1)/2) = (k^2-1)/8$. The following linear ordering T requires no more arc reversings than this number:

$$(4.7) \quad 1 \prec 3 \prec 5 \prec \dots \prec k-2 \prec k \prec 2 \prec 4 \prec 6 \prec \dots \prec k-3 \prec k-1.$$

Finally, notice that \mathcal{D}_k and \mathcal{E}_k are dual relations. \square

Proof of Theorem 4.2. Let G be the ridge defined by inequality (4.2), written more compactly as

$$(4.8) \quad \langle c, x \rangle \leq \frac{k-1}{2} \quad \text{for} \quad \sum_{i=1}^k (x_{i, i \oplus 1} + x_{i \oplus 1, i}) - \sum_{i=1}^k (x_{i, i \oplus 2} + x_{i \oplus 2, i}) \leq \frac{k-1}{2}.$$

We know that G also satisfies (4.3), which we summarize as

$$(4.9) \quad \langle d, x \rangle = 0 \quad \text{for} \quad \sum_{ij \in \mathcal{E}_k} x_{ij} - \sum_{ij \in \mathcal{D}_k} x_{ij} = 0.$$

Then

$$(4.10) \quad \langle 2(k-1)c + 4d, x \rangle \leq (k-1)^2$$

is inequality (4.6) up to a factor 4. There remains to show that inequality (4.10) is valid for any vertex of P_{WO}^n and that it becomes an equality at some vertex of P_{WO}^n not in G .

Let W be a weak order on n . By a *class segment* we mean a subset S of k of the form $\{i, i \oplus 1, \dots, i \oplus \ell\}$ such that all elements of S belong to a same class of W , and, moreover, S cannot be extended in k while keeping this double condition. The *length* of S is $\ell + 1$. (Thus k itself may be a class segment, and its length is by convention k .) Now denote by s the number of class segments of length strictly greater than 1 and by r the number of class segments of odd length. It is not difficult to check $\langle c, x^W \rangle = s$.

We now prove

$$(4.11) \quad 2(k-1)\langle c, x^W \rangle + 4\langle d, x^W \rangle \leq (k-1)^2.$$

Only coordinates x_{ij}^W with $i, j \in k$ appear in inequality (4.11); thus we may assume that any element in $n \setminus k$ is isolated in its equivalence class. Next, if a class segment $S = \{i, i \oplus 1, \dots, i \oplus \ell\}$ has $\ell > 1$, we modify W by breaking only the class C containing S into two successive classes, namely $\{i, i \oplus 1\}$ and $C \setminus \{i, i \oplus 1\}$; this modification does not decrease $\langle c, x^W \rangle$ and leaves $\langle d, x^W \rangle$ unchanged (as easily verified). We may now assume that all class segments of W have length 1 or 2.

If a class C strictly contains a one-element segment $\{i^*\}$, let W^+ and W^- be the weak orders obtained from W by pushing i^* out of its equivalence class one position down and one position up, respectively; that is,

$$W^- = W \setminus \{(j, i^*) \mid j \sim i^* \text{ and } j \neq i^*\},$$

$$W^+ = W \setminus \{(i^*, j) \mid j \sim i^* \text{ and } j \neq i^*\}.$$

(As before, \sim denotes the equivalence relation $W \cap W^{-1}$.) Either $\langle d, x^{W^+} \rangle = \langle d, x^{W^-} \rangle = \langle d, x^W \rangle$ or $\langle d, x^{W^+} \rangle - \langle d, x^W \rangle = \langle d, x^W \rangle - \langle d, x^{W^-} \rangle \neq 0$; thus we may extract i^* from its equivalence class and get another weak order, also called W , without decreasing either $\langle d, x^W \rangle$ or $\langle c, x^W \rangle$. We may now further assume that any element i^* isolated in its segment is also isolated in its class.

In the evaluation of $\langle d, x^W \rangle$, all pairs of W touching one class segment of length 2 have a total contribution zero. Thus, we need look only at the linear ordering T induced by W on the set R of the r elements of k which are alone in their classes.

Notice that \mathcal{D}_k induces on R a relation which is naturally isomorphic to \mathcal{D}_r . Thus using Lemma 4.3 we get

$$\begin{aligned} \langle d, x^W \rangle &= |T \cap \mathcal{E}_r| - |T \cap \mathcal{D}_r| \\ &= |T| - 2|T \cap \mathcal{D}_r| \\ &\leq \frac{r(r-1)}{2} - 2 \cdot \frac{r^2-1}{8} \\ &= \frac{(r-1)^2}{4}. \end{aligned}$$

Hence, as $s = (k-r)/2$,

$$\begin{aligned} 2(k-1)\langle c, x^W \rangle + 4\langle d, x^W \rangle &\leq 2(k-1)\frac{k-r}{2} + 4\frac{(r-1)^2}{4} \\ &= \left(r - \frac{k+1}{2}\right)^2 + \frac{3(k-1)^2}{4}. \end{aligned}$$

As k is odd, we cannot have $r = 0$. For r varying in k , the last expression has maximum value $(k-1)^2$, attained for $r = 1$ or $r = k$.

We have thus established that inequality (4.10) defines a face of P_{WO}^n which contains the ridge G . A vertex x^T in this face but not in G is obtained for T , the linear ordering specified in the proof of Lemma 4.3.

To get the other facet of P_{WO}^n which contains the ridge G , we apply the linear permutation mapping (x_{ij}) onto (x_{ji}) ; this permutation stabilizes both P_{WO}^n and G . \square

5. Lifting the 2-chorded path and 2-chorded wheel inequalities. Again, by making use of our fundamental tool (Proposition 2.2), we infer additional facets of P_{WO}^n from two families of facets of P_{PA}^n .

A *2-chorded path inequality* for P_{PA}^n is obtained as follows (under a specific choice of labels for the elements in n). Let $1, 2, \dots, \ell-1$ be (a path) in n , and let ℓ be an additional element in n . Grötschel and Wakabayashi [9] prove that the inequality

$$(5.1) \quad \sum_{i=1}^{\ell-2} x_{\{i,i+1\}} - \sum_{i=1}^{\ell-3} x_{\{i,i+2\}} + \sum_{\substack{j=2 \\ j \text{ even}}}^{\ell-2} x_{\{j\ell\}} - \sum_{\substack{j=1 \\ j \text{ odd}}}^{\ell-1} x_{\{j\ell\}} \leq \frac{\ell-2}{2}$$

is facet-defining for P_{PA}^n when ℓ is even and at least 4.

THEOREM 5.1. *The inequality lifted from inequality (5.1),*

$$(5.2) \quad \begin{aligned} &\sum_{i=1}^{\ell-2} (x_{i,i+1} + x_{i+1,i}) - \sum_{i=1}^{\ell-3} (x_{i,i+2} + x_{i+2,i}) \\ &+ \sum_{\substack{j=2 \\ j \text{ even}}}^{\ell-2} (x_{j\ell} + x_{\ell j}) - \sum_{\substack{j=1 \\ j \text{ odd}}}^{\ell-1} (x_{j\ell} + x_{\ell j}) \leq \frac{\ell-2}{2}, \end{aligned}$$

defines a facet of P_{WO}^n for ℓ even, $\ell \geq 4$.

Proof. Because of Proposition 2.2, it suffices to show that $\text{permsub}(F)$, for F the facet of P_{PA}^n defined by inequality (5.1), contains all transposition vectors $\text{transp}(u, v)$ for u, v distinct in n . To this aim, we notice that vertex y^E belongs to F at least when E is an equivalence relation with the following equivalence classes, where $i, j \in \{1, 2, \dots, \ell-1\}$ and $i < j$:

(i) for i and j odd,

$$(5.3) \quad \begin{aligned} & \{1, 2\}, \{3, 4\}, \dots, \{i-2, i-1\}, \\ & \{i\}, \\ & \{i+1, \ell\}, \\ & \{i+2, i+3\}, \{i+4, i+5\}, \dots, \{j-2, j-1\}, \\ & \{j\}, \\ & \{j+1, j+2\}, \{j+3, j+4\}, \dots, \{\ell-2, \ell-1\}, \\ & \{\ell+1\}, \{\ell+2\}, \dots, \{n\}; \end{aligned}$$

(ii) for i even and j odd,

$$(5.4) \quad \begin{aligned} & \{1, 2\}, \{3, 4\}, \dots, \{i-1, i\}, \dots, \{j-2, j-1\}, \\ & \{j\}, \\ & \{j+1, j+2\}, \{j+3, j+4\}, \dots, \{\ell-2, \ell-1\}, \\ & \{\ell\}, \{\ell+1\}, \dots, \{n\}; \end{aligned}$$

(iii) for i odd and j even,

$$(5.5) \quad \begin{aligned} & \{1, 2\}, \{3, 4\}, \dots, \{i-2, i-1\}, \\ & \{i\}, \\ & \{i+1, i+2\}, \{i+3, i+4\}, \dots, \{j, j+1\}, \dots, \{\ell-2, \ell-1\}, \\ & \{\ell\}, \{\ell+1\}, \dots, \{n\}; \end{aligned}$$

(iv) for i and j even,

$$(5.6) \quad \begin{aligned} & \{1, 2\}, \{3, 4\}, \dots, \{i-1, i\}, \dots, \{j-1, j\}, \dots, \{\ell-3, \ell-2\}, \\ & \{\ell-1\}, \{\ell\}, \dots, \{n\}. \end{aligned}$$

By (i), we get $\text{transp}(i, j) \in \text{permsub}(F)$ when i and j are odd. For i even and j odd, (ii) gives $\text{transp}(\{i-1, i\}, \{j\}) \in \text{permsub}(F)$, and by subtracting $\text{transp}(i-1, j)$ (which was just shown to belong to $\text{permsub}(F)$), we get $\text{transp}(i, j) \in \text{permsub}(F)$. The cases (i odd, j even) and (i, j even) follow in a similar way from (iii) and (iv). The other transposition vectors $\text{transp}(u, v)$, for $u, v \in \mathfrak{n}$ with $u \neq v$, also belong to $\text{permsub}(F)$. \square

With an adequate relabeling of the elements in \mathfrak{n} , a *2-chorded even wheel inequality* for P_{PA}^n , with $1 \leq k \leq n-1$ and k even, is written as

$$(5.7) \quad \sum_{i=1}^k x_{\{i, i \oplus 1\}} - \sum_{i=1}^k x_{\{i, i \oplus 2\}} + \sum_{\substack{j=1 \\ j \text{ even}}}^k x_{\{jn\}} - \sum_{\substack{j=1 \\ j \text{ odd}}}^k x_{\{jn\}} \leq \frac{k}{2}$$

(where i and j are taken modulo k in $\{1, 2, \dots, k\}$). According to [9], this inequality defines a facet of P_{PA}^n when k is even and $k \geq 8$.

THEOREM 5.2. *The inequality lifted from inequality (5.7) gives a facet-defining inequality for P_{WO}^n when k is even and $k \geq 8$, which reads*

$$(5.8) \quad \begin{aligned} & \sum_{i=1}^k (x_{i, i \oplus 1} + x_{i \oplus 1, i}) - \sum_{i=1}^k (x_{i, i \oplus 2} + x_{i \oplus 2, i}) \\ & - \sum_{\substack{j=1 \\ j \text{ even}}}^k (x_{jn} + x_{nj}) - \sum_{\substack{j=1 \\ j \text{ odd}}}^k (x_{jn} + x_{nj}) \leq \frac{k}{2}. \end{aligned}$$

Proof. As the arguments are similar to those in the previous proof, we just list one example of a useful partition of n ; the corresponding equivalence relation E provides a vertex y^E of P_{PA}^n which belongs to the facet defined by inequality (5.7). For i, j odd in $1, 2, \dots, k$ with $j \notin \{i \ominus 2, i, i \oplus 2\}$ (the case $j \in \{i \ominus 2, i \oplus 2\}$ requires a slight modification), we list the classes of E :

$$(5.9) \quad \begin{aligned} & \{i\}, \\ & \{i \oplus 2, i \oplus 3\}, \{i \oplus 4, i \oplus 5\}, \dots, \{j \ominus 2, j \ominus 1\}, \\ & \{j\}, \\ & \{j \oplus 2, j \oplus 3\}, \{j \oplus 4, j \oplus 5\}, \dots, \{k \ominus 1, k\}, \\ & \{1, 2\}, \{3, 4\}, \dots, \{i \ominus 2, i \ominus 1\}, \\ & \{i \oplus 1, j \oplus 1, n\}, \\ & \{k + 1\}, \{k + 2\}, \dots, \{n - 1\}. \end{aligned}$$

This partition and similar ones obtained by cyclically rotating $1, 2, \dots, k$ help in showing that $\text{permsub}(F)$ contains all transposition vectors $\text{transp}(u, v)$, where F is the facet of P_{PA}^n defined by inequality (5.7), and $1 \leq u < v \leq n$. \square

Grötschel and Wakabayashi [9] present still one more facet-defining inequality for P_{PA}^n , an isolated example which is not “symmetric,” and has some coefficients equal to 2. We leave it to the reader to verify that this inequality is lifted into an additional facet-defining inequality for P_{WO}^n .

Acknowledgments. The authors thank Olivier Hudry for directions to the bibliography on tournaments, and the two referees for comments on the first version and for the mention of [1], [3].

REFERENCES

- [1] H.-J. BANDELT, M. OOSTEN, J. H. G. C. RUTTEN, AND F. C. R. SPIEKSMAN, *Lifting theorems and facet characterization for a class of clique partitioning inequalities*, Oper. Res. Lett., 24 (1999), pp. 235–243.
- [2] J.-C. BERMOND, *Ordres à distance minimum d'un tournoi et graphes partiels sans circuits maximaux*, Math. Sci. Humaines, 37 (1972), pp. 5–25.
- [3] G. BOLOTASHVILI, M. KOVALEV, AND E. GIRLICH, *New facets of the linear ordering polytope*, SIAM J. Discrete Math., 12 (1999), pp. 326–336.
- [4] S. FIORINI, *Determining the automorphism group of the linear ordering polytope*, Discrete Appl. Math., 112 (2001), pp. 121–128.
- [5] S. FIORINI, *Polyhedral Combinatorics of Order Polytopes*, Ph.D. thesis, Université Libre de Bruxelles, Brussels, Belgium, 2001.
- [6] S. FIORINI AND P. FISHBURN, *Weak order polytopes*, submitted.
- [7] P. FISHBURN, *Induced binary probabilities and the linear ordering polytope: A status report*, Math. Social Sci., 23 (1992), pp. 67–80.
- [8] M. GRÖTSCHEL, M. JÜNGER, AND G. REINELT, *Facets of the linear ordering polytope*, Math. Programming, 33 (1985), pp. 43–60.
- [9] M. GRÖTSCHEL AND Y. WAKABAYASHI, *Facets of the clique partitioning polytope*, Math. Programming, 47 (1990), pp. 367–387.
- [10] M. GURGEL, *Poliedros de Grafos Transitivos*, Ph.D. thesis, Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, Brazil, 1992.
- [11] M. GURGEL AND Y. WAKABAYASHI, *The Complete Pre-order Polytope: Facets and Separation Problem*, manuscript, 1996.
- [12] M. GURGEL AND Y. WAKABAYASHI, *Adjacency of vertices of the complete pre-order polytope*, Discrete Math., 175 (1997), pp. 163–172.
- [13] J.-F. LASLIER, *Tournament Solutions and Majority Voting*, Stud. Econom. Theory 7, Springer, Berlin, 1997.

EFFICIENCY OF LOCAL SEARCH WITH MULTIPLE LOCAL OPTIMA*

JOSSELIN GARNIER[†] AND LEILA KALLEL[‡]

Abstract. The first contribution of this paper is a theoretical investigation of combinatorial optimization problems. Their landscapes are specified by the set of neighborhoods of all points of the search space. The aim of the paper consists of the estimation of the number N of local optima and the distributions of the sizes (α_j) of their attraction basins. For different types of landscapes we give precise estimates of the size of the random sample that ensures that at least one point lies in each attraction basin.

A practical methodology is then proposed for identifying these quantities (N and (α_j) distributions) for an unknown landscape, given a random sample of starting points and a local steepest ascent search. This methodology can be applied to any landscape specified with a modification operator and provides bounds on search complexity to detect all local optima. Experiments demonstrate the efficiency of this methodology for guiding the choice of modification operators, eventually leading to the design of problem-dependent optimization heuristics.

Key words. combinatorial complexity, local search, neighborhood graph, randomized starting solution

AMS subject classifications. 68R99, 90C10, 60F99

PII. S0895480199355225

1. Introduction. In the field of stochastic optimization, two search techniques have been widely investigated during the last decade: simulated annealing [26] and evolutionary algorithms (EAs) [7, 8]. These algorithms are now widely recognized as methods of order zero for function optimization as they impose no condition on function regularity. However, the efficiency of these search algorithms, in terms of the time they require to reach the solution, is strongly dependent on the choice of the modification operators used to explore the landscape. These operators in turn determine the neighborhood relation of the landscape under optimization.

This paper provides a new methodology allowing one to estimate the number and the sizes of the attraction basins of a landscape specified in relation to some modification operator. This allows one to derive bounds on the probability that one samples a point in the basin of the global optimum, for example. Further, this method could be used for guiding the choice of efficient problem-dependent modification operators or representations.

Formally, a landscape can be denoted by $\mathcal{L} = (f, \mu, E)$, where f is the function to optimize and μ the modification operator that is applied to elements of the search space E . The structure of the landscape heavily depends on the choice of the modification operators, which in turn may depend on the choice of the representation (the coding of the candidate solutions into binary or gray strings, for example). Hence, before the optimization process can be started, there is a number of practical choices (representation and operators) that determine the landscape structure. Consequently,

*Received by the editors April 13, 1999; accepted for publication (in revised form) December 10, 2001; published electronically January 16, 2002.

<http://www.siam.org/journals/sidma/15-1/35522.html>

[†]Laboratoire de Statistique et Probabilités, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex 4, France (garnier@cict.fr)

[‡]Centre de Mathématiques Appliquées, Ecole Polytechnique, 91128 Palaiseau Cedex, France (kallel@cmapx.polytechnique.fr).

these choices are often crucial for the success of stochastic search algorithms.

Some research has studied how the fitness landscape structure impacts the potential search difficulties [14, 22, 23, 27]. It is shown that every complex fitness landscape can be represented as an expansion of elementary landscapes—one term in the Fourier expansion—which are easier to search in most cases. This result has been applied to solve a difficult NP-complete problem [21] (the identification of a minimal finite k -state automaton for a given input-output behavior), using evolutionary algorithms. Other theoretical studies of search feasibility consider the whole landscape as a tree of local optima, with a label describing the depth of the attraction basin at each node [17, 20]. Such a construction naturally describes the inclusion of the local attraction basins present in the landscape. These studies investigate tree structures that ensure a minimal correlation between the strength of the local optima and their proximity to the global optimum, with respect to an ultrametric distance on the tree. However, from a practical point of view, the tree describing the repartition of local optima is unknown and too expensive in terms of computational cost to determine for a given landscape.

The lack of an efficient method at reasonable cost that allows one to characterize a given landscape motivates the construction of heuristics for extracting a priori statistical information about landscape difficulty, for example, based on random sampling of the search space. We cite from the field of evolutionary algorithms: Fitness Distance relations, first proposed in [9] and successfully used to choose problem dependent random initialization procedures [12, 15]; Fitness Improvement of evolution operators, first proposed in [6], then extended and successfully used to choose binary crossover operators [13] and representations [10]. However, even if such heuristics can guide the a priori choice of some EA parameters, they do not give significant information about landscape structure; for instance, recent work suggests that very different landscapes (leading to different EA behaviors) can share the same fitness distance relation [19, 11]. Further, the efficiency of such summary statistics is limited to the sampled regions of the space and therefore does not necessarily help the long term convergence results as implicitly illustrated in [13], for example. This gives strong motivation for developing tools that allow one to derive more global (beyond the sampled regions) information on the landscape at hand, relying on an implicit assumption of stationarity of the landscape. Along that line, this paper proposes a new method to identify the number and the repartition of local optima with respect to a given neighborhood relation of a given landscape. The proposed method applies to any neighborhood relation specified with a modification operator and hence provides a practical tool to compare landscapes obtained with different operators and representations.

The framework is the following. We assume that the search space E can be split into the partition E_1, \dots, E_N of subspaces which are attraction basins of local maxima m_1, \dots, m_N of the fitness function. We also assume that there exists a local search algorithm (for example, a steepest ascent) which is able to find from any point of the search space the corresponding local maximum:

$$\Theta : \begin{cases} E & \rightarrow \{m_1, \dots, m_N\}, \\ x & \mapsto m_j \text{ if } x \in E_j. \end{cases}$$

The basic problem consists of detecting all local maxima m_j . This is equivalent to finding a way to put a point in all attraction basins E_j because the local search

algorithm will complete the job. We shall develop the following strategy. First, we shall study the direct problem, which consists of studying the covering of the search space by a collection of points randomly distributed when the partition (E_j) is known. Second, we shall deal with the inverse problem which consists of estimating the number of local maxima from information deduced from the covering.

Direct problem (section 4). One puts M points randomly in the search space. The question is the following: Given the statistical distribution of the relative sizes of the attraction basins and their number N , what is the probability $p_{N,M}$ that at least one point lies in every attraction basin? This probability is very important. Indeed, using the local search algorithm, it is exactly equal to the probability of detecting all local maxima of the function.

Inverse problem (section 5). The statistical distribution of the relative sizes of the attraction basins and their number are assumed to be known for computing $p_{N,M}$ in section 4. Unfortunately, this is rarely the case in practical situations, and one wants to estimate both. The strategy is to put randomly M initial points in the search space and to detect the corresponding local maxima by the local search algorithm. The data we collect is the set $(\beta_j)_{j \geq 1}$ of the number of maxima detected with j initial points. Of course, β_0 is unknown (number of local maxima of the landscape that have not been detected). The question is the following: How can the total number of local maxima $N = \sum_{j=0}^{\infty} \beta_j$ be efficiently estimated from the set $(\beta_j)_{j \geq 1}$? A lower bound is $\bar{N} = \sum_{j=1}^{\infty} \beta_j$, but we aim at constructing a better estimator.

The paper is divided into three parts. First, section 4 addresses the direct problem of sample sizing in the case of basins of random sizes, then in the case of basins of equal sizes. Second, section 5 is devoted to the estimation of the distribution of the relative basins sizes for an unknown landscape, using a random sample from the search space. This is achieved by a two step methodology: section 5.2 starts by considering a parametrized family of laws for the relative sizes of basins for which it derives the corresponding covering of the search space (law of (β_j)). Then section 5.3 comments on how these results can be practically used for characterizing the sizes of basins of an unknown landscape. For instance, it proposes to compare the covering of an unknown landscape (given by the empirically observed (β_j) values) to the coverings studied in section 5.2. Finally, the last part of the paper (section 6) is devoted to some experiments that validate (section 6.1) and illustrate (section 6.2) the methodology: First, a landscape is purposely designed to test the reliability of the method according to the size of the random sample and to the number of local optima. (Recall that the theoretical results are asymptotic with respect to N and M .) Second, the method is used to investigate some problems known to be difficult to optimize for EAs. For each problem, we also compare the landscapes related to different mutation operators.

2. Notations and definitions. Consider a fitness $f : E \rightarrow \mathbb{R}$, and a neighborhood relation induced by a modification operator μ , such that the number of different μ -neighbors (neighbors that can be obtained by one application of μ to x) of $x \in E$ is “bounded.” In the following, we denote by N the number of local optima of \mathcal{L} and by (α_j) the random variables describing the sizes of the attraction basins of \mathcal{L} (normalized to the average size). As shown in [24, 25], a local improvement algorithm is efficient to find quickly a local optimum starting from some given point. Among the possible algorithms we present the Steepest Ascent (SA), also called the optimal adjacency algorithm in [24].

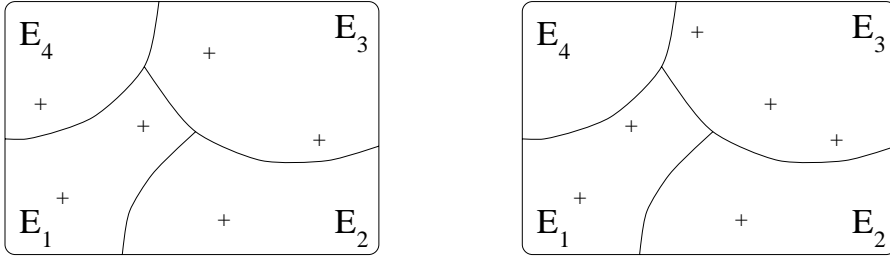


FIG. 1. Schematic representations of the search space E with $N = 4$ attraction basins. $M = 6$ points have been randomly placed on both pictures. As a result there is at least one point in each attraction basin in the left picture, but not in the right picture, where E_4 is empty.

STEEPEST ASCENT ALGORITHM (SA).

Input: A fitness $f : E \rightarrow \mathbb{R}$, an operator μ , and a point $X \in E$.

Algorithm: Modify X by repeatedly performing the following steps:

- Record for all μ -neighbors of X , denoted by $\mu^i(X) : (i, f(\mu^i(X)))$.
- Assign $X = \mu^i(X)$, where i is chosen such that $f(\mu^i(X))$ reaches the highest possible value. (This is the steepest ascent.)
- Stop when no strictly positive improvement in μ -neighbors' fitnesses has been found.

Output: The point X , denoted by $\mu^{SA}(X)$.

The SA algorithm thus consists of selecting the best neighbors after the entire neighborhood is examined. An alternative algorithm, the so-called First Improvement (FI), consists of accepting the first favorable neighbor as soon as it is found, without further searching. Note that in the FI case there are extra free parameters which are the order in which the neighborhood is searched. As pointed out in [16, p. 470], the steepest ascent is often not worth the extra computation time, although it is sometimes much quicker. Nevertheless, our focus in this paper is not a complete optimization of the computational time, so we let this problem remain an open question.

DEFINITION 2.1. *Attraction basin:* The attraction basin of a local optimum m_j is the set of points X_1, \dots, X_k of the search space such that a steepest ascent algorithm starting from X_i ($1 \leq i \leq k$) ends at the local optimum m_j . The normalized size of the attraction basin of the local optimum m_j is then equal to $k/|E|$.

Remarks. 1. This definition of the attraction basins yields a partition of the search space into different attraction basins, as illustrated in Figure 1. The approach proposed in this paper is based on this representation of the search space into a partition of attraction basins and could be generalized to partitions defined with alternative definitions of attraction basins.

2. In the presence of local constancy in the landscape, the above definition of the steepest ascent (and hence also the related definition of the attraction basins) is not rigorous. For instance, if the fittest neighbors of point p have the same fitness value, then the steepest ascent algorithm at point p has to make a random or user-defined choice. Nevertheless, even in the presence of local constancy, the comparison of the results (distribution of (α_j)) obtained with different steepest ascent choices may give useful information about the landscape and guide the best elitism strategy: “move” to fitter points or “move” to strictly fitter points only.

3. Summary of the results. Given a distribution of (α_j) , we determine M_{min} , the minimal size of a random sample of the search space, in order to sample at least one

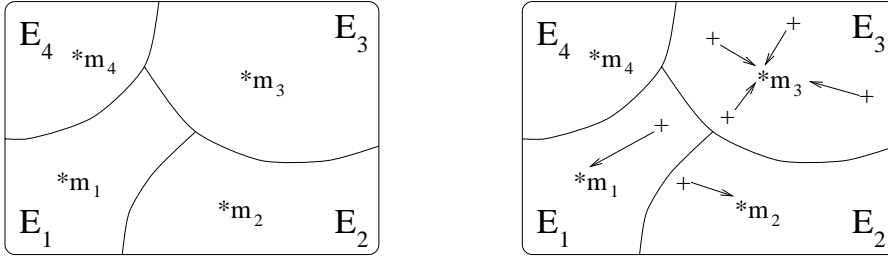


FIG. 2. Schematic representation of the search space E with its $N = 4$ attraction basins and the 4 corresponding local maxima m_1, \dots, m_4 . In the left picture we have put $M = 6$ points randomly chosen. We apply the search algorithm and detect 3 maxima according to the right picture, so that we have $\beta_1 = 2$, $\beta_2 = 0$, $\beta_3 = 0$, $\beta_4 = 1$, and $\beta_j = 0$ for $j \geq 5$.

point in each attraction basin of the landscape. Two particular cases are investigated.

1. *Deterministic configuration.* All the attraction basins have the same size ((α_j) are deterministic).
2. *Random configuration.* The sizes of the attraction basins are completely random ((α_j) are uniformly distributed).

In both configurations, we give the value of M_{min} as a function of the number of local optima N . For instance, a random sample of size $M_{min} = N(\ln N + \ln a)$ for the deterministic configuration (resp., $M_{min} = aN^2$ for the random configuration) ensures that a point is sampled in each attraction basin with probability $\exp(-1/a)$.

We then address the inverse problem of identifying the distribution of the normalized sizes $(\alpha_j)_{j=1, \dots, N}$ of the attraction basins for an unknown landscape. Some direct analysis is first required as discussed below.

Direct analysis. Consider a random sample $(X_i)_{i=1, \dots, M}$ uniformly chosen in the search space. For each $i \in \{1, \dots, M\}$, a steepest ascent starting from X_i (with the modification operator(s) at hand μ) ends at the local optimum $\mu^{SA}(X_i)$. Define β_j as the number of local optima (m_j) that are reached by exactly j points from (X_i) (see an example in Figure 2):

$$\beta_j = \text{card}\{k; \text{card}\{i; \mu^{SA}(X_i) = m_k\} = j\}.$$

Proposition 5.1 gives the distribution of (β_j) for a family of parametrized distributions Law_γ for (α_j) asymptotically with respect to N and M . More precisely, let $(Z_j)_{j=1, \dots, N}$ be a family of positive real-valued independent random variables with Gamma distributions whose densities are

$$p_\gamma(z) = \frac{\gamma^\gamma}{\Gamma(\gamma)} z^{\gamma-1} e^{-\gamma z}.$$

Suppose $\alpha_j = \frac{Z_j}{\sum_{i=1}^N Z_i}$. Let $a > 0$ be a constant. If $N \gg 1$ and $M = [aN]$,¹ then the expected number $\beta_{j,\gamma} := \mathbb{E}_\gamma[\beta_j]$ is

$$\beta_{j,\gamma} = N \frac{\Gamma(j+\gamma)}{j! \Gamma(\gamma)} \frac{a^j \gamma^\gamma}{(a+\gamma)^{j+\gamma}} + o(N).$$

¹ $[x]$ stands for the integral part of the real number x .

Moreover, the ratio $r = M/N$ is the unique solution of

$$(3.1) \quad \frac{\sum_{j=1}^{\infty} \beta_{j,\gamma}}{M} = \frac{1 - (1 + \frac{r}{\gamma})^{-\gamma}}{r}.$$

The latter equation is then used to find a good estimator of N , with observed values of the variables β_j , as explained below.

Inverse problem. Given an unknown landscape, we then propose to characterize the distribution of (α_j) through the empirical estimation of the distribution of the random family (β_j) . In fact, by construction, the distribution of (α_j) and that of (β_j) are tightly related: We experimentally determine observed values taken by (β_j) (random sampling and steepest ascent search). Then, for each γ value, we use a χ^2 test to compare the observed law for (β_j) to the law β should (theoretically) obey if the law of (α_j) were Law_γ . Naturally, we find a (possible) law for (α_j) if and only if one of the latter tests is positive. Otherwise, we gain only the knowledge that (α_j) does not obey the law Law_γ . Note also that the method can be used to determine subparts of the search space with a given distribution for (α_j) . In case the law of (α_j) obeys Law_γ , (3.1) is used to find a good estimator of N .

Last, section 6 validates the methodology of section 5, by considering known landscapes with random and deterministic sizes of basins, showing that the estimations of the number of local optima N are accurate, even if M is much smaller than N . Further, we apply the methodology on unknown landscapes and show that the Hamming binary and gray F1 landscapes contain much more local optima than the 3-bit-flip landscapes.

4. Direct problem. We assume that the search space E can be split into the partition E_1, \dots, E_N of subspaces which are attraction basins of local maxima m_1, \dots, m_N of the fitness function. Let us put a sample of M points randomly in the search space. We aim at computing the probabilities $p_{N,M}$ that at least one point of the random sample lies in each attraction basin.

PROPOSITION 4.1. *If we denote by $\alpha_j := |E_j|/|E|$ the normalized size of the j th attraction basin, then*

$$(4.1) \quad p_{N,M} = \sum_{k=0}^N (-1)^k \sum_{1 \leq j_1 < \dots < j_k \leq N} (1 - \alpha_{j_1} - \dots - \alpha_{j_k})^M.$$

Proof. Let us denote by A_j the event

$$A_j = \{\text{there is no point in } E_j\}.$$

The probability of the intersection of a collection of events A_j is easy to compute. For any $1 \leq i_1 < \dots < i_k \leq N$, if there is one initial point chosen uniformly in E ($M = 1$), then we have

$$\mathbb{P}(A_{i_1} \cap \dots \cap A_{i_k}) = 1 - \alpha_{i_1} - \dots - \alpha_{i_k}.$$

If there are M initial points chosen uniformly and independently in E , then

$$\mathbb{P}(A_{i_1} \cap \dots \cap A_{i_k}) = (1 - \alpha_{i_1} - \dots - \alpha_{i_k})^M.$$

On the other hand, $1 - p_{N,M}$ is the probability that at least one of the attraction basin contains no point, which reads as

$$p_{N,M} = 1 - \mathbb{P}(A_1 \cup \dots \cup A_N).$$

The result follows from the inclusion-exclusion formula [29, Formula 1.4.1a]. \square

Proposition 4.1 gives an exact expression for $p_{N,M}$ which holds true for whatever N , M , and (α_j) are but is quite complicated. The following corollaries show that the expression of $p_{N,M}$ is much simpler in some particular configurations.

COROLLARY 4.2. 1. *If the attraction basins all have the same size $\alpha_j \equiv 1/N$ (the so-called D -configuration), then*

$$p_{N,M} = \sum_{k=0}^N C_N^k (-1)^k (1 - k/N)^M.$$

2. *If, moreover, the numbers of attractors and initial points are large $N \gg 1$ and $M = [N(\ln N + \ln a)]$, where $a > 0$ is a constant, then*

$$p_{N,M} \xrightarrow{N \rightarrow \infty} \exp(-a^{-1}).$$

3. *Let us denote by M_D the number of points which are necessary to detect all local maxima. Then in the asymptotic framework $N \gg 1$, M_D obeys the distribution of*

$$M_D = N \ln N - N \ln Z + o(N),$$

where Z is an exponential variable with mean 1.

An exponential variable with mean 1 is a random variable whose density with respect to the Lebesgue measure over \mathbb{R}^+ is $p(z) = \exp(-z)$.

Proof. The first point is a straightforward application of Proposition 4.1. It is actually referenced in the literature as the coupon-collector's problem [4]. The fact that $M_D/(N \ln N)$ converges in probability to 1 is also well known. The statistical distribution of $M_d - N \ln N$ is given by [4, Chapter 2, Example 6.6] which establishes the second point. The third point then follows readily from the identity $\mathbb{P}(M_d \geq m) = 1 - p_{N,m-1}$. \square

COROLLARY 4.3. 1. *If the sizes of the attraction basins are random (the so-called R -configuration), in the sense that their joint distribution is uniform over the simplex of \mathbb{R}^N ,*

$$S_N := \left\{ \alpha_i \geq 0, \sum_{i=1}^N \alpha_i = 1 \right\},$$

and the numbers of attractors and initial points are large: $N \gg 1$ and $M = [N^2 a]$, $a > 0$, then

$$p_{N,M} \xrightarrow{N \rightarrow \infty} \exp(-a^{-1}).$$

2. *Let us denote by M_R the number of points which are necessary to detect all local maxima. Then in the asymptotic framework $N \gg 1$, M_R obeys the distribution of*

$$M_R = N^2 Z^{-1} + o(N^2),$$

where Z is an exponential variable with mean 1.

A construction of the R -configuration is the following. Assume that the search space E is the interval $[0, 1)$. Choose $N - 1$ points $(a_i)_{i=1, \dots, N-1}$ uniformly over $[0, 1]$

and independently. Consider the order statistics $(a_{\sigma(i)})_{i=1,\dots,N-1}$ of this sample, that is to say, permute the indices of these points so that $a_{\sigma(0)} := 0 \leq a_{\sigma(1)} \leq \dots \leq a_{\sigma(N-1)} \leq a_{\sigma(N)} := 1$. Denote the spacings by $\alpha_j = a_{\sigma(j)} - a_{\sigma(j-1)}$ for $j = 1, \dots, N$. Note that α_j is also called the j th coverage. If the j th attraction basin E_j is the interval $[a_{\sigma(j-1)}, a_{\sigma(j)})$, then the sizes $(\alpha_j)_{j=1,\dots,N}$ of the attraction basins $(E_j)_{j=1,\dots,N}$ obey a uniform distribution over the simplex S_N .

Proof. From (4.1) and the relation $\sum_{j=1}^N \alpha_j = 1$ we get that

$$(4.2) \quad p_{N,M} = \sum_{k=0}^{\infty} (-1)^k p_{N,M,k},$$

$$(4.3) \quad p_{N,M,k} = \sum_{1 \leq j_1 < \dots < j_k \leq N} \mathbb{E} [(1 - \alpha_{j_1} - \dots - \alpha_{j_k})^M],$$

where \mathbb{E} stands for the expectation with respect to $(\alpha_j)_{j=1,\dots,N}$ whose distribution is uniform over S_N . As pointed out in [29, section 9.6a], the probability distribution of the sum of any k of the N coverages α_j is described by the repartition function given by [29, Formula 9.6.1], which shows that it admits a density $q_{N,k}(\alpha)$ with respect to the Lebesgue measure over $[0, 1]$:

$$q_{N,k}(\alpha) = \frac{(N-1)!}{(k-1)!(N-k-1)!} \alpha^{k-1} (1-\alpha)^{N-1-k}.$$

We can thus write a closed form expression for $p_{N,M,k}$:

$$(4.4) \quad \begin{aligned} p_{N,M,k} &= C_N^k \int_0^1 d\alpha q_{N,k}(\alpha) (1-\alpha)^M \\ &= C_N^k \frac{(N-1)!}{(k-1)!(N-k-1)!} \int_0^1 d\alpha \alpha^{k-1} (1-\alpha)^{M+N-1-k} \end{aligned}$$

$$(4.5) \quad = \frac{1}{k!} \frac{N!}{(N-k)!} \frac{(N-1)!}{(N-1-k)!} \frac{(M+N-1-k)!}{(M+N-1)!}.$$

We shall first prove an estimate of $p_{N,M,k}$.

Step 1. $p_{N,M,k} \leq \frac{1}{k!} \left(\frac{N^2}{M}\right)^k$. We have $N!/(N-k)! \leq N^k$ and $(N-1)!/(N-k-1)! \leq (N-1)^k \leq N^k$. For any $k = 0, \dots, N$ we also have $(M+N-1-k)!/(M+N-1)! \leq M^{-k}$. Substituting these inequalities into (4.5) establishes the desired estimate.

Step 2. For any fixed k , if $M = [aN^2]$, then $p_{N,M,k} \rightarrow a^{-k}/k!$ as $N \rightarrow \infty$. On the one hand, $C_N^k \frac{(N-1)!}{(k-1)!(N-k-1)!} \times N^{-2k} \rightarrow 1/(k!(k-1)!)$. On the other hand,

$$N^{2k} \times \int_0^1 d\alpha \alpha^{k-1} (1-\alpha)^{[aN^2]+N-1-k} = \int_0^1 ds s^{k-1} \left(1 - \frac{s}{N^2}\right)^{[aN^2]+N-1-k}.$$

Since $\left(1 - \frac{s}{N^2}\right)^{[aN^2]+N-1-k}$ is bounded by 1 and converges to $\exp(-as)$ as $N \rightarrow \infty$, the dominated convergence theorem implies that

$$N^{2k} \times \int_0^1 d\alpha \alpha^{k-1} (1-\alpha)^{[aN^2]+N-1-k} \xrightarrow{N \rightarrow \infty} (k-1)! a^{-k},$$

which yields the result by (4.4).

Step 3. Convergence of $p_{N,M}$ when $M = \lfloor aN^2 \rfloor$.

We first choose some $K \geq 1$. We have from the result of Step 1

$$(4.6) \quad \left| p_{N,M} - \sum_{k=0}^K (-1)^k p_{N,M,k} \right| \leq \sum_{k=K+1}^{\infty} \frac{1}{k!} \left(\frac{N^2}{\lfloor aN^2 \rfloor} \right)^k.$$

Substituting the result of Step 2 into (4.6) shows that

$$\limsup_{N \rightarrow \infty} \left| p_{N,M} - \sum_{k=0}^K \frac{(-1)^k a^{-k}}{k!} \right| \leq \sum_{k=K+1}^{\infty} \frac{a^{-k}}{k!}.$$

This inequality holds true for any K , so letting $K \rightarrow \infty$ completes the proof of the corollary. \square

It follows from the corollaries that about $N \ln N$ points are needed in the D-configuration to detect all maxima, while about N^2 points are needed to expect the same result in the R-configuration. This is due to the fact that there exist very small attraction basins in the R-configuration. Actually, it can be proved that the smallest attraction basin in the R-configuration has a relative size which obeys an exponential distribution with mean N^{-2} . (For more detail about the asymptotic distribution concerning order statistics, we refer the reader to [29].) That is why a number of points of the order of N^2 is required to detect this very small basin.

Mean values. The expected value of M_D is

$$\mathbb{E}[M_D] = N \ln N + NC + o(N),$$

where C is the Euler's constant whose value is $C \simeq 0.58$. The expected value of M_R/N^2 is equal to infinity. This is due to the fact that the tail corresponding to exceptional large values of M_R is very important:

$$\mathbb{P}(M_R \geq N^2 a) \xrightarrow{N \rightarrow \infty} 1 - \exp(-a^{-1}) \underset{a \gg 1}{\simeq} a^{-1}.$$

Standard deviations. The normalized standard deviation, which is equal to the standard deviation divided by the mean, of the number of points necessary to detect all local maxima in the D-configuration is equal to

$$\sigma_D := \frac{\sqrt{\mathbb{E}[M_D^2] - \mathbb{E}[M_D]^2}}{\mathbb{E}[M_D]} \underset{N \rightarrow \infty}{\simeq} \frac{\pi}{\sqrt{6}(\ln N + C)},$$

which goes to 0 as $N \rightarrow \infty$, which proves in particular that $M_D/(N \ln N)$ converges to 1 in probability. This is, of course, not surprising. The D-configuration has a deterministic environment, since all basins have a fixed size, so that we can expect an asymptotic deterministic behavior. The situation is very different in the R-configuration which has a random environment, and it may happen that the smallest attraction basin be much smaller than its expected size N^{-2} . That is why the fluctuations of M_D , and especially the tail corresponding to exceptional large values, are very important.

5. Inverse problem.

5.1. Formulation of the problem. We now focus on the inverse problem. We look for the number N of local maxima of the fitness function and also some pieces of information on the distribution of the sizes of the corresponding attraction basins. We assume that we can use an algorithm that is able to associate to any point of the search space the corresponding local maximum. In order to detect all local maxima, we should apply the algorithm to every point of the search space. Nevertheless, this procedure is far too long since the search space has a large cardinality. Practically, we shall apply the algorithm to M points that will be chosen randomly in the search space E . The result of the search process can consequently be summed up by the following set of observed values ($j \geq 1$):

$$(5.1) \quad \beta_j := \text{number of maxima detected with } j \text{ points.}$$

Our arguments are based upon the following observations. First note that $\bar{N} := \sum_{j=1}^{\infty} \beta_j$ is the number of detected maxima. It is consequently a lower bound of the total number of local maxima N , but a very rough estimate, in the sense that it may happen that many maxima are not detected, especially those whose attraction basins are small. Besides, \bar{N} represents less information than the complete set $(\beta_j)_{j \geq 1}$. By a clever treatment of this information, we should be able to find a better estimate of N than \bar{N} .

5.2. Analysis. The key point is that the distribution of the set β_j is closely related to the distribution of the sizes of attraction basins. Let us assume that the relative sizes $(\alpha_j)_{j=1, \dots, N}$ of the attraction basins can be described by a distribution parametrized by some positive number γ as follows. Let $(Z_j)_{j=1, \dots, N}$ be a sequence of independent random variables whose common distribution has density p_γ with respect to the Lebesgue measure over $(0, \infty)$:²

$$(5.2) \quad p_\gamma(z) = \frac{\gamma^\gamma}{\Gamma(\gamma)} z^{\gamma-1} e^{-\gamma z},$$

where Γ is the Euler's Gamma function $\Gamma(s) := \int_0^\infty e^{-t} t^{s-1} dt$. The density p_γ is plotted in Figure 3. It is the so-called Gamma density with parameters (γ, γ) [5, p. 47, Formula 2.2]. Under p_γ , the expected value of Z_1 is 1 and its standard deviation is $1/\sqrt{\gamma}$. In the following we shall say that we are under H^γ if the relative sizes of the attraction basins $(\alpha_j)_{j=1, \dots, N}$ can be described as $(Z_1/T_N, \dots, Z_N/T_N)$, where $T_N := \sum_{j=1}^N Z_j$ and the distribution of Z_j has density p_γ . Note that the large deviations principle (Cramer's theorem [1, Chapter 1]) applied to the sequence (Z_j) yields that for any $x > 0$ there exists $c_{\gamma, x} > 0$ such that ³

$$(5.3) \quad \mathbb{P}_\gamma \left(\left| \frac{T_N}{N} - 1 \right| \geq x \right) \leq \exp(-N c_{\gamma, x}),$$

which shows that, in the asymptotic framework $N \gg 1$, the ratio Z_j/N stands for the relative size α_j up to a negligible correction. The so-called D- and R-configurations described in section 4 are particular cases of this general framework:

- For $\gamma = \infty$, $Z_j \equiv 1$, and $T_N = N$, so that we get back the deterministic D-configuration.

²If γ is a positive integer, then p_γ is a negative-binomial distribution.

³Applying the procedure described in [1] establishes that $c_{\gamma, x} = \gamma(x - 1 - \ln x)$.

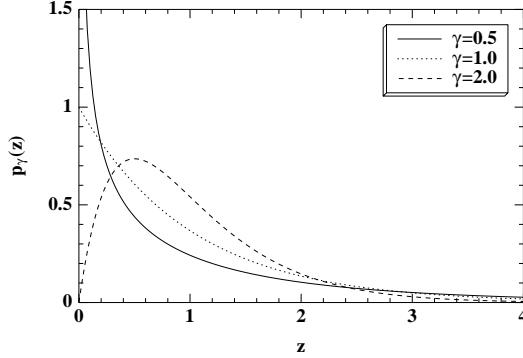


FIG. 3. Probability density of the sizes of the attraction basins under H^γ for different γ .

- For $\gamma = 1$, the Z_j 's obey independent exponential distributions with mean 1, and the family $\alpha_j = Z_j/T_N$ obeys the uniform distribution over S_N [18].

The important statement is the following one.

PROPOSITION 5.1. *Under H^γ the expected values $\beta_{j,\gamma} := \mathbb{E}_\gamma[\beta_j]$ of the β_j 's can be computed for any N , M , and $j \leq M$:*

$$(5.4) \quad \beta_{j,\gamma} = N \frac{\Gamma(\gamma + j)}{j! \Gamma(\gamma)} \frac{\Gamma(N\gamma)}{\Gamma((N-1)\gamma)} \frac{\Gamma((N-1)\gamma + M - j)}{\Gamma(N\gamma + M)} \frac{M!}{(M-j)!}.$$

In the asymptotic framework $N \gg 1$, if $M = [aN]$, then $\beta_{j,\gamma}$ can be expanded as

$$(5.5) \quad \beta_{j,\gamma} = N \frac{\Gamma(j + \gamma)}{j! \Gamma(\gamma)} \frac{a^j \gamma^\gamma}{(a + \gamma)^{j+\gamma}} + o(N).$$

Proof. Under H^γ , the probability that j of the M points lie in the k th attraction basin can be computed explicitly:

$$\mathbb{P}_\gamma(j \text{ points in } E_k) = C_M^j \mathbb{E}_\gamma \left[\alpha_k^j (1 - \alpha_k)^{M-j} \right],$$

where $\alpha_k = Z_k / \sum_i Z_i$ and \mathbb{E}_γ stands for the expectation of Z_j with distribution p_γ . Accordingly, in terms of the Z_i 's this expression reads

$$\mathbb{P}_\gamma(j \text{ points in } E_k) = C_M^j \mathbb{E}_\gamma \left[\frac{Z_k^j \check{Z}_k^{M-j}}{(Z_k + \check{Z}_k)^M} \right],$$

where $\check{Z}_k = \sum_{i \neq k} Z_i$. The random variables Z_k and \check{Z}_k are independent. The probability density of Z_k is p_γ given by (5.2). The random variable \check{Z}_k is the sum of $N-1$ independent random variables with densities p_γ , so that its probability density is [5, p. 47, Formula 2.3]:

$$\check{p}_\gamma(\check{z}) = \frac{\gamma^{(N-1)\gamma}}{\Gamma((N-1)\gamma)} \check{z}^{(N-1)\gamma-1} e^{-\gamma \check{z}}.$$

Accordingly,

$$\mathbb{P}_\gamma(j \text{ points in } E_k) = C_M^j \int_0^\infty dz \int_0^\infty d\check{z} p_\gamma(z) \check{p}_\gamma(\check{z}) \frac{z^j \check{z}^{M-j}}{(z + \check{z})^M}.$$

By the change of variables $u = z/(z + \check{z})$ and $v = z + \check{z}$ we get

$$\mathbb{P}_\gamma(j \text{ points in } E_k) = C_M^j \int_0^1 du \int_0^\infty dv v^{N\gamma-1} e^{-\gamma v} u^{\gamma+j-1} (1-u)^{(N-1)\gamma+M-j-1}.$$

The integral with respect to v is straightforward by definition of the Gamma function. The integral with respect to u can be obtained via tabulated formulae [5, p. 47, Formula 2.5]. This gives the explicit formula (5.4) for $\beta_{j,\gamma}$ since

$$\mathbb{E}_\gamma[\beta_j] = \sum_{k=1}^N \mathbb{P}_\gamma(j \text{ points in } E_k).$$

If $N \gg 1$ and $M = [aN]$, then we have $N^{-\gamma} \times \Gamma(N\gamma)/\Gamma((N-1)\gamma) \rightarrow \gamma^\gamma$, $N^{\gamma+j} \times \Gamma((N-1)\gamma + M - j)/\Gamma(N\gamma + M) \rightarrow 1/(\gamma + a)^{j+\gamma}$, and $N^{-j} \times M!/(M-j)! \rightarrow a^j$ as $N \rightarrow \infty$. This proves the asymptotic formula (5.5). \square

In particular, the distribution of the β_j 's under the D-configuration is Poisson in the asymptotic framework $N \gg 1$:

$$\beta_{j,\infty} = N e^{-M/N} \frac{1}{j!} \left(\frac{M}{N}\right)^j + o(N),$$

while it is geometric under the R-configuration:

$$\beta_{j,1} = N \frac{1}{1 + \frac{M}{N}} \left(\frac{\frac{M}{N}}{1 + \frac{M}{N}}\right)^j + o(N).$$

From (5.5) we can deduce that the following relation is satisfied by the ratio $r = M/N$:

$$(5.6) \quad \frac{\sum_{j=1}^\infty \beta_{j,\gamma}}{M} = \frac{1 - (1 + \frac{r}{\gamma})^{-\gamma}}{r}.$$

5.3. Estimator of the number of local maxima. We now have sufficient tools to exhibit a good estimator of the number of local maxima. We remind the reader of the problem at hand. We assume that some algorithm is available to determine from any given point the corresponding local maximum. We choose randomly M points in the search space and detect the corresponding local maxima. We thus obtain a set of values $(\beta_j)_{j \geq 1}$ as defined by (5.1). We can then determine from the set of values $(\beta_j)_{j \geq 1}$ which configuration H^{γ_0} is the most probable, or at least which H^{γ_0} is the closest configuration of the real underlying distribution of the relative sizes of the attraction basins. The statistics used to compare observed and expected results is the so-called χ^2 goodness of fit test [28, section 8.10], which consists first of calculating for each γ

$$T_\gamma := \sum_{j \in \Omega} \frac{(\beta_j - \beta_{j,\gamma})^2}{\beta_{j,\gamma}},$$

where $\Omega := \{j \in \mathbb{N}, \beta_j \geq 1\}$ is the set of the indices j for which $\beta_j \geq 1$. Obviously, a large value for T_γ indicates that the corresponding $\beta_{j,\gamma}$ are far from the observed ones; that is to say, H^γ is unlikely to hold. Conversely, the smaller T_γ , the more likely H^γ holds true. In order to determine the significance of various values of T_γ , we need the

distribution of the statistics. A general result states that if the hypothesis H^{γ_0} does hold true, then the distribution of T_{γ_0} is approximatively the so-called χ^2 -distribution with degrees of freedom equal to the cardinality of the set Ω minus 1. Consequently, we can say that the closest configuration of the real underlying distribution of the relative sizes of the attraction basins is H^{γ_0} , where γ_0 is given by

$$(5.7) \quad \gamma_0 = \operatorname{argmin} \{T_\gamma, \gamma > 0\}.$$

Furthermore, we can estimate the accuracy of the configuration H^{γ_0} by referring T_{γ_0} to tables of the χ^2 -distribution for $\operatorname{card}(\Omega) - 1$ degrees of freedom. A value of T_{γ_0} much larger than the one indicated in the tables means that none of the configurations H^γ hold true. Nevertheless, H^{γ_0} is the closest distribution of the real one.

Remark. The distribution theory of χ^2 goodness of fit statistic can be found in [3, Chapter 30]. The result is in any case approximate, and all the poorer as they are many expected $\beta_{j,\gamma}$ less than five. These cases must be avoided by combining cells. However, power in the tail regions is then lost, where differences are more likely to show up.

Defining γ_0 as (5.7), we denote by $\bar{\beta}$ the quantity

$$\bar{\beta} = \frac{\sum_{j=1}^{\infty} \beta_j}{M}.$$

From (5.6), under H^{γ_0} the ratio $\alpha = M/N$ is the unique solution of

$$(5.8) \quad \bar{\beta} = \frac{1 - (1 + \frac{r}{\gamma_0})^{-\gamma_0}}{r}.$$

Consequently, once we have determined γ_0 , formula (5.8) is a good estimator of the ratio $\alpha = M/N$, hence N .

6. Experiments. Given a landscape \mathcal{L} , the following steps are performed in order to identify a possible law for the number and sizes of the attraction basins of \mathcal{L} , among the family of laws Law_γ studied above.

1. Choose a random sample $(X_i)_{i=1,\dots,M}$ uniformly in E .
2. Perform a steepest ascent starting from each X_i up to $\mu^{SA}(X_i)$.
3. Compute β_j defined as the number of local optima reached by exactly j initial points X_i .
4. Compare the observed law of β to the laws of $\beta(\gamma)$ for different γ values, using the χ^2 test.

To visualize the comparison of the last item, we propose to plot the obtained χ^2 value for different γ values. We also plot the corresponding χ^2 value below which the test is positive with a confidence of 95%.

6.1. Experimental validation. The results obtained in section 5 are asymptotic with respect to the number of local optima N and the size of the random sample M . Hence, before the methodology can be applied, some experimental validation is required in order to determine practical values for M and N for which the method is reliable. This is achieved by applying the methodology to determine the distribution of (α_j) (normalized sizes of the attraction basins) in two known purposely constructed landscapes: The first contains basins with random sizes; the second contains basins with equal sizes.

Results are plotted in Figures 4, 5, and 6. Samples with smaller sizes than those shown in these figures yield β_j values which are not rich enough to allow a significant

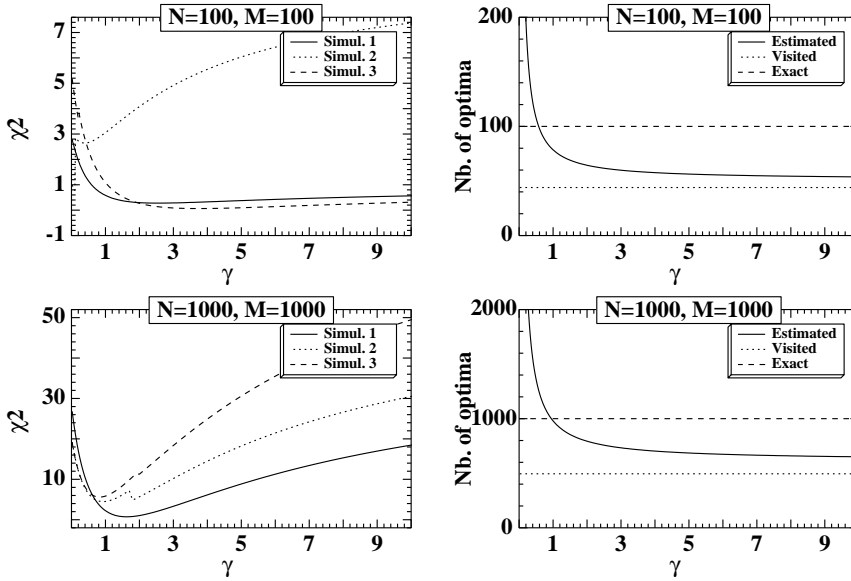


FIG. 4. Basins with random uniform sizes: The left figures plot the χ^2 test results (i.e., the values of T_γ), comparing the empirically observed β distribution to the family of γ -parametrized distributions. The right figures plot for different γ values the estimation of the number of local optima computed by (5.8). These estimations are very robust (only one estimation is plotted) and are accurate for $\gamma_0 = 1$. The same figures also show the visited numbers of optima actually visited by the steepest ascent ($\bar{N} = \sum_{j=1}^{\infty} \beta_j$). The numerical simulations exhibit unstable results for the χ^2 test for small N values and $M = N$.

χ^2 test comparison. For instance, the χ^2 test requires that observed β_j are nonnull for some $j > 1$ at least. (Some initial points are sampled in the same attraction basin.) In case all initial points are sampled in different attraction basins the χ^2 test comparison is not significant.

These experiments give practical bounds on the sample sizes (in relation to the number of local optima) for which the methodology is reliable: The numerical simulations exhibit unstable results for the χ^2 test for $M = N$ and small N values (Figures 4). When N increases and M is bounded ($M \leq \min(2000, 3N)$ in the experiments), results become stable and accurate (Figures 5). Further, we demonstrate that the estimation of number of local optima is accurate, even when initial points visit a small number of attraction basins of the landscape (Figure 6). This situation is even more striking in the experiments of the following section on Baluja F1 problems.

6.2. The methodology at work. Having seen that the methodology is a powerful tool, provided that the information obtained for β is rich enough, we apply it to investigate the landscape structure of the difficult gray- and binary-coded F1 Baluja problems [2] for a 1-bit-flip and 3-bit-flip neighborhood relations.

Gray-coded Baluja F1 functions. Consider the function of k variables (x_0, \dots, x_{k-1}) , with $x_i \in [-2.56, 2.56]$ [2]:

$$F1(\vec{x}) = \frac{100}{10^{-5} + \sum_{i=0}^{k-1} |y_i|}, \quad y_0 = x_0, \quad \text{and } y_i = x_i + y_{i-1} \quad \text{for } i = 1, \dots, k-1.$$

It reaches its maximum value among 10^7 at point $(0, \dots, 0)$. The gray-encoded F1g and binary F1b versions, with, respectively, 2, 3, and 4 variables encoded on 9 bits,

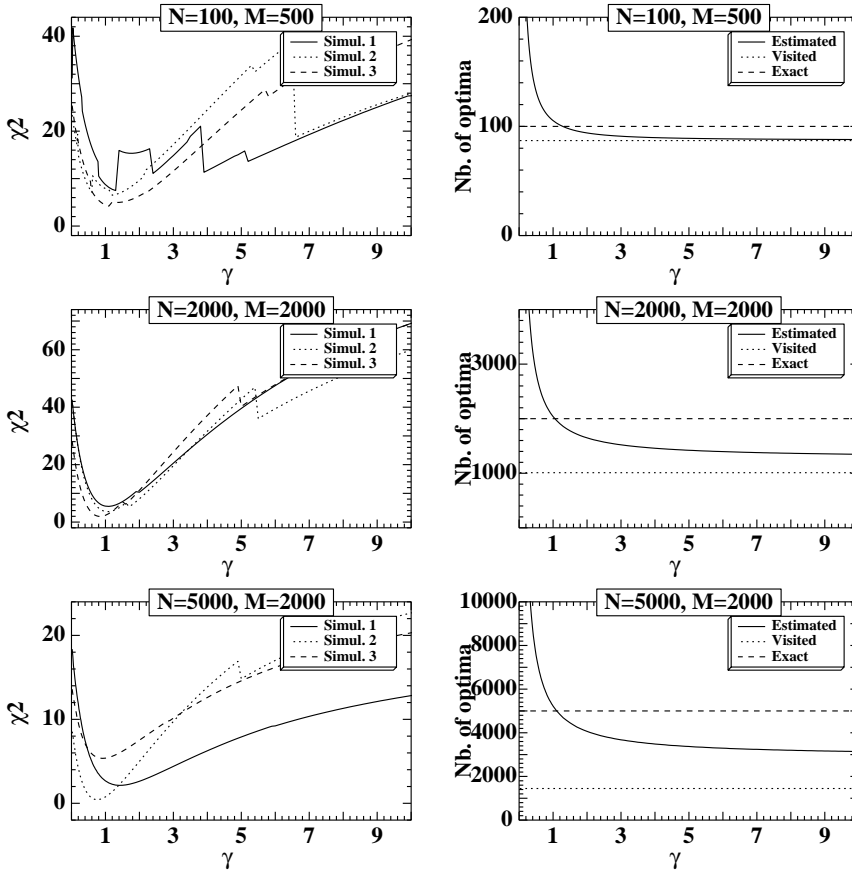


FIG. 5. The same as in Figure 4 with different values for N and M . Stable results are obtained when N increases and M is bounded ($M \leq \min(2000, 3N)$ here). The estimation of N corresponding to the smallest χ^2 value ($\gamma = 1$) is very accurate.

are considered. This encoding consequently corresponds to the binary search space with $l = 9k$.

Considering the 1-bit-flip mutation (Hamming landscape), Figure 7 shows that the distribution of the sizes of the basins is closer to the random configuration than to the deterministic one and that the estimated number of local optima is similar for the binary and gray codings. On the other hand, considering the 3-bit-flip mutation (Figure 8), the estimated number of local optima drops significantly for both problems: less than 250 for both binary and gray landscapes, whereas the Hamming landscape contains thousands of local optima (Figure 7).

Experiments at problem sizes $l = 18$ and $l = 36$ have been carried out in addition to the plotted ones ($l = 27$), leading to similar results for both F1g and F1b problems: The number of local optima of the 3-bit-flip landscape is significantly smaller than that of the Hamming landscape. For example, when $l = 18$, there are less than 10 local optima in the 3-bit-flip landscape versus hundreds in the Hamming landscape. When $l = 36$, the estimations for the Hamming landscape show about two times more local optima for the gray than for the binary encoding (resp., 45000 and 25000). Still for $l = 36$, but for the 3-bit-flip landscape, the estimated number of local optima

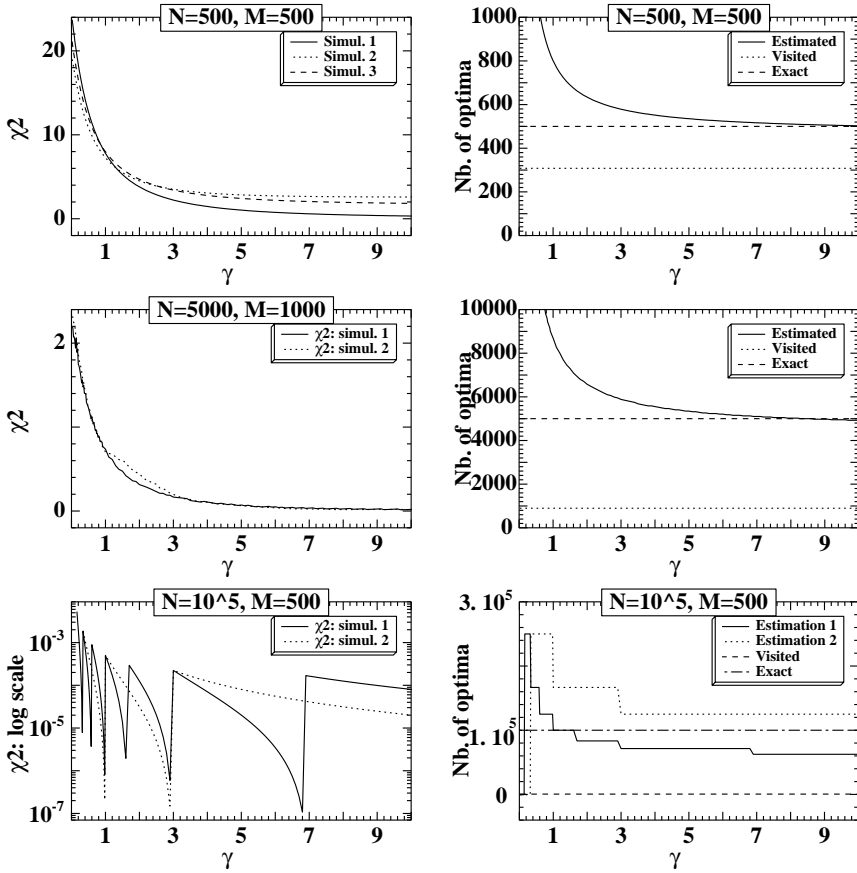


FIG. 6. Basins with deterministic equal sizes: The χ^2 results are stable for smaller sample sizes than those of the random configuration. The bottom figures correspond to the case $N = 10^5$ and $M = 500$, where the χ^2 test is not significant, yet the predicted number of local optima is very accurate! With 500 initial points, 497 local optima have been visited, while there are actually 10^5 optima. Yet, formula (5.8) is able to estimate the true number with an error of less than 30% when the adequate γ value is used.

drops, respectively, to 1400 and 1000.

A new optimization heuristic. A simple search strategy for solving difficult problems naturally follows from the methodology presented in this paper: Once the number N and distribution of the attraction basins is estimated following the guidelines summarized in the beginning of section 6, generate a random sample whose size is set to $N(\ln N + \ln a)$ if the sizes of the basins are close to the deterministic configuration (resp., aN^2 if the sizes of the basins are close to random). Then a simple steepest ascent starting from each point of the sample ensures that the global optimum is found with probability $\exp(-1/a)$.

In the 27-bits F1 problem, this heuristic demonstrates to be very robust and efficient in solving the problem with the 3-bit-flip operator. Using a 3-bit-flip mutation steepest ascent, an initial random sample of 5 points (versus 100 with 1-bit-flip mutation) is enough to ensure that one point at least lies in the global attraction basin (experiments based on 50 runs)! This is due to the fact that the basin of the global optimum is larger than the basins of the other local optima. In order to detect all

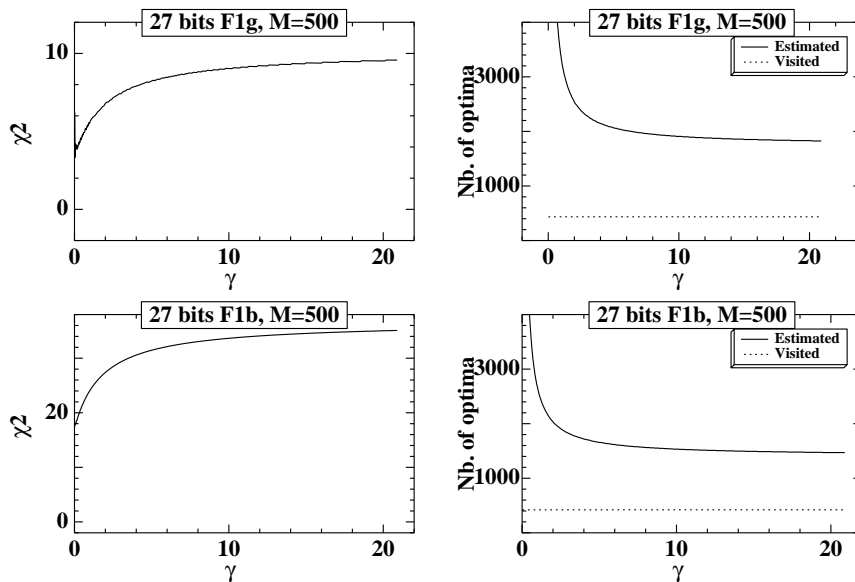


FIG. 7. The difficult Baluja 27-bits F1 gray (F1g) and binary (F1b) landscapes with a 1-bit-flip mutation. Experiments with samples of sizes $M = 2000$ and $M = 5000$ show the same results for the χ^2 test, and the corresponding estimations of the number of local maxima converge to a stable value around 4000.

attraction basins, we can estimate the required sample size to 62500 (250×250 using Corollary 4.3 and the estimation of $N = 250$ in the experiments of Figure 8).

6.3. Discussion. This paper provides a new methodology allowing one to estimate the number and the sizes of the attraction basins of a landscape specified in relation to some modification operator. This allows one to derive bounds on the probability that one samples a point in the basin of the global optimum, for example. Further, it allows one to compare landscapes related to different modification operators or representations, as illustrated with the Baluja problem.

The efficiency of the proposed method is certainly dependent on the class of laws of (α_j) (sizes of the attraction basins) for which the distribution of β is known. We have chosen a very particular family of distributions p_γ for representing all possible distributions for the relative sizes of attraction basins. The constraints for this choice are twofold and contradictory. On the one hand, a large family of distributions is required to be sure that at least one of them is sufficiently close to the observed repartition (β_j) . On the other hand, if we choose an overlarge family, then we need a lot of parameters to characterize the distributions. It is then very difficult to estimate all parameters and consequently to decide which distribution is the closest to the observed one. That is why the choice of the family is very delicate and crucial. We feel that the particular family p_γ that has been chosen (5.2) fulfills determinant conditions. First, it contains two very natural distributions, the so-called D- and R-configurations that we have studied with great detail. Second, it is characterized by a single parameter easy to estimate. Third, it contains distributions with a complete range of variances, from 0 (the D-configuration) to infinity, by going through 1 (the R-configuration).

However, the experiments with the Baluja problem appeal for refining the class

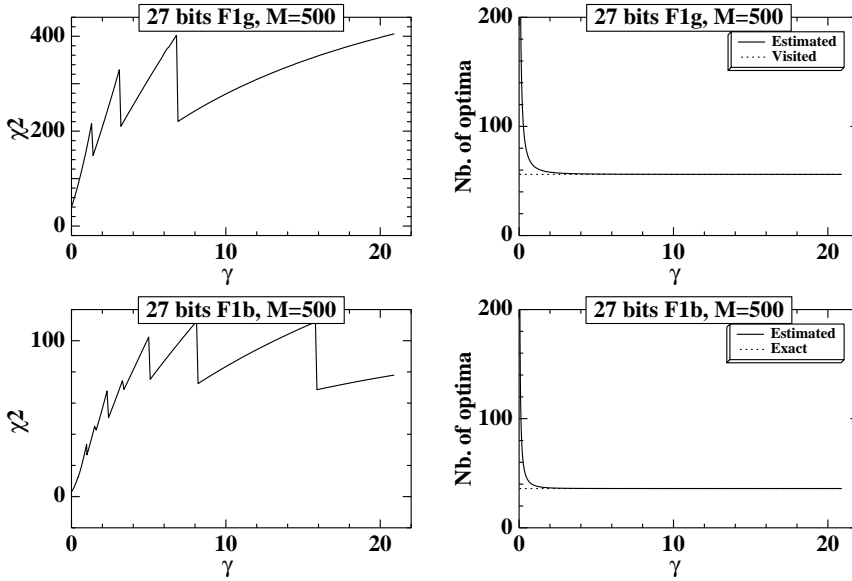


FIG. 8. The difficult Baluja 27-bits F1 gray (F1g) and binary (F1b) landscapes with a 3-bit-flip mutation: the number of local optima drops significantly compared to the Hamming 1-bit-flip landscape. These results are confirmed by experiments using samples of sizes $M = 2000$ and $M = 5000$ which give the same estimation for the number of local optima.

of laws of (α_j) around basins with random sizes. We may propose $\alpha_j = Z_j / \sum_{i=1}^N Z_i$, where Z_j are independent and identically distributed with one of the distributions of the bidimensional family $p_{\gamma,\delta}(\cdot)$, $\gamma > 0$, $\delta > 0$:

$$p_{\gamma,\delta}(z) = \frac{\gamma^\delta}{\Gamma(\delta)} z^{\delta-1} e^{-\gamma z}.$$

The parameter δ characterizes the distribution of the sizes of the small basins, since $p_{\gamma,\delta}(z) \sim z^{\delta-1}$ as $z \rightarrow 0$, while γ characterizes the distribution of the sizes of the large basins, since the decay of $p_{\gamma,\delta}(z)$ as $z \rightarrow \infty$ is essentially governed by $e^{-\gamma z}$. The density $p_{\gamma,\delta}$ is the so-called Gamma density with parameters (γ, δ) [5, p. 47, Formula 2.2]. This family presents more diversity than the family $p_\gamma(\cdot)$ we have considered in section 5.2. The expected value of β_j is under $p_{\gamma,\delta}$:

$$\beta_{j,\gamma,\delta} = N \frac{\Gamma(j+\delta)}{j! \Gamma(\delta)} \frac{a^j \gamma^\delta}{(a+\gamma)^{j+\delta}} \Big|_{a=M/N} + o(N).$$

The method of estimating the number of local minima described in section 5.3 could then be applied with this family.

To apply our method we have also made a crucial choice which consists of executing the local search algorithm from randomly distributed points. We do so because we have no a priori information on the landscape at hand. However, assume for a while that we have some a priori information about the fitness function, e.g., its average value. Consequently, we could hope that starting with points whose fitnesses are better than average would improve the detection of the local maxima. Nevertheless, extensive computer investigations of some particular problems have shown that this is

not the case [16, p. 456], possibly because a completely random sampling of starting points allows one to get a wider sample of local optima.

A first application of the methodology presented in this paper is to compare landscapes obtained when different operators are used (k -bit-flip binary mutations for different k values, for example). However, the complexity of this method is directly related to size of the neighborhood of a given point. Hence, its practical usefulness to study k -bit-flip landscapes is limited when k value increases. Hence, it seems most suited to investigate different representations. Its extension to nonbinary representations is straightforward, provided that a search algorithm that leads to the corresponding local optimum can be provided for each representation. Further, this methodology can be used to determine subparts of the search space, such that (α_j) obey a particular law, hence guiding a hierarchical search in different subparts of the space.

Note finally that the distributions of the sizes of basins do not fully characterize landscape difficulty. Depending on the relative position of the attraction basins, the search still may range from easy to difficult. Additional information is necessary to compare landscapes difficulty. Further work may address such issues to extract additional significant information in order to guide the choice or the design of problem dependent operators and representations.

Acknowledgment. The authors would like to thank the referee for making useful comments that have improved the presentation of this paper.

REFERENCES

- [1] R. AZENCOTT, *Grandes déviations et applications*, in Proceedings of Ecole d'Eté de Probabilités de Saint-Flour, P. L. Hennequin, ed., Springer-Verlag, Berlin, 1980, pp. 1–176.
- [2] S. BALUJA, *An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics*, Technical Report CMU-CS-95-193, Carnegie Mellon University, Pittsburgh, PA, 1995.
- [3] H. CRAMER, *Mathematical Methods of Statistics*, Princeton University Press, Princeton, NJ, 1946.
- [4] R. DURRETT, *Probability: Theory and Applications*, Wadsworth and Brooks/Cole, Pacific Grove, CA, 1991.
- [5] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Wiley, New York, 1971.
- [6] D. B. FOGEL AND A. GHOZEIL, *Using fitness distributions to design more efficient evolutionary computations*, in Proceedings of the Third IEEE International Conference on Evolutionary Computation, T. Fukuda, ed., IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 11–19.
- [7] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, MA, 1989.
- [8] J. H. HOLLAND, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.
- [9] T. JONES AND S. FORREST, *Fitness distance correlation as a measure of problem difficulty for genetic algorithms*, in Proceedings of the 6th International Conference on Genetic Algorithms, L. J. Eshelman, ed., Morgan Kaufmann, San Francisco, CA, 1995, pp. 184–192.
- [10] L. KALLEL, *Convergence des algorithmes génétiques : aspects spatiaux et temporels*, Ph.D. thesis, Ecole Polytechnique, Palaiseau, France, 1998.
- [11] L. KALLEL, B. NAUDTS, AND M. SCHOENAUER, *On functions with a fixed fitness distance relation*, in Proceedings of the 1999 Congress on Evolutionary Computation (ICEC'99), IEEE, 1999, pp. 1910–1916.
- [12] L. KALLEL AND M. SCHOENAUER, *Alternative random initialization in genetic algorithms*, in Proceedings of the 7th International Conference on Genetic Algorithms, Th. Bäck, ed., Morgan Kaufmann, San Francisco, CA, 1997, pp. 268–275.

- [13] L. KALLEL AND M. SCHOENAUER, *A priori predictions of operator efficiency*, in Artificial Evolution'97, Lecture Notes in Comput. Sci. 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, eds., Springer-Verlag, Berlin, 1997.
- [14] B. MANDERICK, M. DE WEGER, AND P. SPIESSENS, *The genetic algorithm and the structure of the fitness landscape*, in Proceedings of the 4th International Conference on Genetic Algorithms, R. K. Belew and L. B. Booker, eds., Morgan Kaufmann, San Francisco, CA, 1991, pp. 143–150.
- [15] P. MERZ AND B. FREISLEBEN, *Memetic algorithms and the fitness landscape of the graph bipartitioning problem*, in Proceedings of the 4th Conference on Parallel Problems Solving from Nature, Lecture Notes in Comput. Sci. 1498, A. E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, eds., Springer-Verlag, Berlin, 1998, pp. 765–774.
- [16] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [17] G. PARISI, M. MEZARD, AND M. A. VIRASORO, *Spin Glass Theory and Beyond*, World Scientific, Teaneck, NJ, 1987.
- [18] R. PYKE, *Spacings*, J. Roy. Statist. Soc. Ser. B, 27 (1965), pp. 395–449.
- [19] R. J. QUICK, V. J. RAYWARD-SMITH, AND G. D. SMITH, *Fitness distance correlation and ridge functions*, in Proceedings of the 4th Conference on Parallel Problems Solving from Nature, Lecture Notes in Comput. Sci. 1498, A. E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, eds., Springer-Verlag, Berlin, 1998, pp. 77–86.
- [20] R. RAMMAL, G. TOULOUSE, AND M. A. VIRASORO, *Ultrametricity for physicists*, Rev. Modern Phys., 58 (1986), pp. 765–788.
- [21] V. SLAVOV AND N. I. NIKOLAEV, *Genetic algorithms, fitness sublandscapes and subpopulations*, in Foundations of Genetic Algorithms 5, W. Banzhaf and C. Reeves, eds., Morgan Kaufmann, San Francisco, CA, 1998.
- [22] P. STADLER, *Complex systems and binary networks*, in Towards a Theory of Landscapes, Springer-Verlag, Berlin, 1995, pp. 77–163.
- [23] P. STADLER, *Landscapes and their correlation functions*, J. Math. Chem., 20 (1996), pp. 1–45.
- [24] C. A. TOVEY, *Local improvement on discrete structures*, in Local Search and Combinatorial Optimization, E. Aarts and J. K. Lenstra, eds., Wiley, New York, 1997, pp. 57–89.
- [25] C. A. TOVEY, *Hill climbing with multiple local optima*, SIAM J. Alg. Disc. Math., 6 (1985), pp. 384–393.
- [26] P. J. VAN LAARHOVEN AND E. H. L. AARTS, *Simulated Annealing: Theory and Applications*, Kluwer Academic Press, Dordrecht, The Netherlands, 1987.
- [27] E. D. WEINBERGER, *Correlated and uncorrelated fitness landscapes and how to tell the difference*, Biological Cybernetics, 63 (1990), pp. 325–336.
- [28] G. B. WETHERILL, *Elementary Statistical Methods*, Chapman and Hall, London, 1972.
- [29] S. S. WILKS, *Mathematical Statistics*, Wiley, New York, 1962.

CYCLIC COLORINGS OF 3-POLYTOPES WITH LARGE MAXIMUM FACE SIZE*

OLEG V. BORODIN[†] AND DOUGLAS R. WOODALL[‡]

Abstract. A cyclic coloring of a plane graph is a vertex coloring in which, for each face f , all the vertices in the boundary of f have different colors. It is proved that if G is a 3-polytope (3-connected plane graph) with maximum face size Δ^* , then G is cyclically $(\Delta^* + 1)$ -colorable provided that Δ^* is large enough. The figure $\Delta^* + 1$ is sharp.

Key words. cyclic coloring, cyclic chromatic number, 3-polytope, plane graph

AMS subject classifications. 05C15, 52B10

PII. S089548019630248X

1. Introduction. The cyclic chromatic number $\chi_c = \chi_c(G)$ of a plane graph G is the smallest number of colors in a vertex-coloring of G in which, for each face f , all the vertices in the boundary of f have different colors. Clearly $\chi_c \geq \Delta^*$, where $\Delta^* = \Delta^*(G)$ is the largest number of vertices in a face; and the pyramid (wheel) with one Δ^* -face and Δ^* triangular faces has $\chi_c = \Delta^* + 1$.

The cyclic chromatic number was introduced explicitly by Ore and Plummer [11], who proved that $\chi_c \leq 2\Delta^*$ whenever $\Delta^* \geq 3$. However, Ringel [13] had earlier conjectured that $\chi_c \leq 6$ if $\Delta^* = 4$, and this was proved by Borodin [2, 3]. It is pointed out in [2, 12] that for each $\Delta^* \geq 4$ there is a 2-connected plane graph with $\chi_c = \lfloor \frac{3}{2}\Delta^* \rfloor$, and it is conjectured that $\chi_c \leq \frac{3}{2}\Delta^*$, which would therefore be best possible. It has recently been proved [5] that $\chi_c \leq \frac{9}{5}\Delta^*$ for all $\Delta^* \geq 3$.

Stronger results hold for 3-polytopes (or, equivalently, for 3-connected planar graphs: Steinitz [14] proved that a (simple) planar graph G is the 1-skeleton of a 3-polytope if and only if G is 3-connected). For 3-polytopes, Plummer and Toft [12] proved (*inter alia*) that $\chi_c \leq \Delta^* + 4$ if $\Delta^* \geq 42$. Borodin [4] improved this to $\chi_c \leq \Delta^* + 3$ if $\Delta^* \geq 24$, and we here prove that $\chi_c \leq \Delta^* + 2$ if $\Delta^* \geq 61$ and $\chi_c \leq \Delta^* + 1$ if $\Delta^* \geq 122$. As mentioned above, the bound $\chi_c \leq \Delta^* + 1$ is best possible, but it would clearly be desirable to reduce the lower bounds of 61 and 122 on Δ^* . Since this paper was first submitted, Enomoto, Horňák and Jendrol' [8] have proved that $\chi_c \leq \Delta^* + 1$ if $\Delta^* \geq 60$, and Horňák and Jendrol' [10] have proved that $\chi_c \leq \Delta^* + 2$ if $\Delta^* \geq 40$. (It is conjectured in [12] that $\chi_c \leq \Delta^* + 2$ for *all* 3-polytopes.) It would also be of interest to characterize 3-polytopes with large Δ^* for which $\chi_c = \Delta^* + 1$; we have not been able to solve this problem. The rest of this paper is devoted to the proof of the following theorem.

*Received by the editors April 22, 1996; accepted for publication (in revised form) January 2, 2002; published electronically February 27, 2002.

<http://www.siam.org/journals/sidma/15-2/30248.html>

[†]Institute of Mathematics, Siberian Branch, Russian Academy of Sciences, Novosibirsk, 630090, Russia (brdnoleg@math.nsc.ru). Part of this work was carried out while this author was visiting Nottingham, funded by Visiting Fellowship Research Grant GR/K00561 from the Engineering and Physical Sciences Research Council. The work of this author was also partly supported by grant 93-01-01486 of the Russian Foundation for Fundamental Research.

[‡]School of Mathematical Sciences, University of Nottingham, Nottingham, NG7 2RD, England (douglas.woodall@nottingham.ac.uk).

THEOREM 1.1. *If G is a 3-polytope with $\Delta^*(G) \leq k$, then*

- (a) $\chi_c(G) \leq k + 1$ if $k \geq 122$,
- (b) $\chi_c(G) \leq k + 2$ if $k \geq 61$.

2. Preliminaries. Let G be a 3-polytope or a 3-connected plane graph; let V , E , and F be the sets of vertices, edges, and faces of G ; and let $v \in V$ and $f \in F$. We say that two vertices *see* each other (*across* a face) if there is a face that is incident with both of them. The *cyclic degree* $d_c(v)$ of v is the number of vertices $w \neq v$ that see v . The *degree* $d(v)$ of v (or $d(f)$ of f) is the number of edges incident with it, and $N(v)$ is the set of vertices adjacent to v , so that $d(v) = |N(v)|$. A k -*vertex* (k -*face*) is a vertex (face) with degree k , and a $\geq k$ -*vertex* has degree at least k (etc.). A vertex v has *type* $(a, \leq b, \geq c)$, or is an $(a, \leq b, \geq c)$ -*vertex*, if its incident faces have degrees (in no particular order) $d_1 = a$, $d_2 \leq b$ and $d_3 \geq c$; other types are defined analogously. A type followed by a superscript \odot describes the incident faces *in cyclic order* around v . Note that, since G is 3-connected, the cyclic degree of a (d_1, \dots, d_k) -vertex is $\sum_{i=1}^k (d_i - 2)$.

If H is a 3-connected graph and $v \in V(H)$, a *contractible v -edge* is an edge e incident with v such that the graph H/e is 3-connected. We shall use the following known result.

LEMMA 2.1. *Let H be a 3-connected graph with at least five vertices, and let v be a vertex such that H has no contractible v -edge. Then*

- (a) (Halín [9]) $d_H(v) \geq 4$ and
- (b) (Ando, Enomoto, and Saito [1]) v has at least three neighbors of degree 3.

The following lemma is fundamental.

LEMMA 2.2. *For fixed positive integers k, h such that there exists a 3-polytope with $\Delta^* \leq k$ that is not cyclically h -colorable, let G be such a 3-polytope with as few vertices as possible. Then $d_c(v) \geq h$ for every vertex v of G .*

Proof. If $h \leq k$, then G is the pyramid (wheel) with one h -face and h triangular faces. Thus G has $h + 1$ vertices all of which see each other so that $d_c(v) = h$ for every vertex v . (We do not need the uniqueness of the pyramid, only its existence, in order to justify the previous sentence.) Thus we may assume that $h \geq k + 1$.

Assuming the result is false, choose a vertex v with minimum degree such that $d_c(v) \leq h - 1$. Add edges if necessary joining consecutive neighbors of v to form a 3-polytope H containing a wheel with vertex-set $\{v\} \cup N(v)$ with v as its central vertex. Note that adding these edges does not increase Δ^* , and any cyclic coloring of $H - v$ will give a cyclic coloring of G if v can be colored appropriately.

Suppose H has a contractible v -edge e . Since $\Delta^*(H/e) \leq \Delta^*(H) \leq \Delta^*(G) \leq k$, and H/e has fewer vertices than G , it follows that H/e has a cyclic h -coloring. Apply this coloring to $G - v$. Since $d_c(v) \leq h - 1$, we can find a color for v that will create a cyclic h -coloring of G , contrary to hypothesis.

Thus H has no contractible v -edge. By Lemma 2.1, $d_G(v) = d_H(v) \geq 4$ and v has a neighbor u such that $d_H(u) = 3$. Both faces incident with edge uv in G must be triangles, since otherwise $d_H(u) > d_G(u) \geq 3$. Thus u is a $(3, 3, \leq k)$ -vertex in G , and $d_c(u) \leq k \leq h - 1$. However, this contradicts the minimum-degree property of v , since $d_G(u) = 3 < d_G(v)$. This contradiction completes the proof of Lemma 2.2. \square

It is proved in [6] that if G is a 3-polytope with $\Delta^*(G) \leq k$, then G has a vertex v with $d_c(v) \leq k + 2$ if $k \geq 21$ (but not if $k \leq 20$). It follows from this and Lemma 2.2 that $\chi_c(G) \leq \Delta^*(G) + 3$ if $\Delta^*(G) \geq 21$. However, this is the best that can be proved

by the use of cyclic degrees alone, since the k -gonal prism (= cylinder) has $\Delta^* = k$ and $d_c(v) = k + 2$ for every vertex v . Instead of using a single vertex of small cyclic degree in proving Theorem 1.1, our strategy must therefore be to use configurations of several vertices.

As is usual with results of this type, our proof involves identifying (in section 3) various configurations that are reducible, that is, that cannot occur in a minimal counterexample, and then using the method of discharging (in sections 4 and 5) to obtain a contradiction to the supposition that a minimal counterexample exists.

For the rest of the paper, we suppose that $G = (V, E, F)$ is a counterexample to Theorem 1.1(a) or (b) that has as few vertices as possible and, subject to this, as few edges as possible. Thus $\Delta^*(G) \leq k$, and either $k \geq 122$ and $\chi_c(G) \geq k + 2$ or $k \geq 61$ and $\chi_c(G) \geq k + 3$. Note that in either case, by Lemma 2.2,

$$(2.1) \quad G \text{ contains no vertex } v \text{ with } d_c(v) \leq k,$$

hence no vertex of type $(3, 3, \leq k)$, $(3, 4, \leq k - 1)$, etc. Our aim is to obtain a contradiction to the existence of G .

3. Structural information. In this section we identify various reducible configurations, that is, configurations that cannot occur in our smallest counterexample G . We shall assume that $k \geq 61$ and that G is minimal such that $\Delta^*(G) \leq k$ and $\chi_c(G) \geq k + 2$; if G is minimal such that $\Delta^*(G) \leq k$ and $\chi_c(G) \geq k + 3$, then the argument is similar but simpler.

LEMMA 3.1. (a) *Let v be a vertex of G that is incident with triangular faces xyv, yvz . Then $d(z) \neq 3$, and the other face f of G incident with edge vz is not a triangle.*

(b) *If uvw is a 4-face such that $d(u) = d(x) = 3$ and $d(w) = 4$, then it is not possible that edges uw, vw , and wx separate uvw from three triangles.*

Proof. (a) Note that $\Delta^*(G - vy) \leq k$, and $\chi_c(G - vy) \geq k + 2$ since every cyclic coloring of $G - vy$ is a cyclic coloring of G . By the minimality of G , $G - vy$ is not a 3-polytope. Thus $\{x, z\}$ is a cutset of $G - vy$. Let C'_1 and C'_2 be the components of $G - \{vy, x, z\}$ containing v and y , respectively, and let $C_1 = C'_1 - v$ and $C_2 = C'_2 - y$. By (2.1), G has no $(3, 3, \leq k)$ -vertices, and so $d(v) \geq 4$ and $d(y) \geq 4$; thus C_1 and C_2 are nonempty, C_1 is a component of $G - \{x, v, z\}$, and C_2 is a component of $G - \{x, y, z\}$. Since G is 3-connected, each of x, z is adjacent to each of C_1, C_2 . Thus $d(z) \geq 4$.

If $f = vzw$ is a triangle, then w and y belong to different components (C_1 and C'_2) of $G - \{x, v, z\}$. The same argument shows that x and z are in different components of $G - \{y, v, w\}$. However, $C_2 \cup \{x, z\}$ induces a connected graph by the previous paragraph. This contradiction proves (a).

(b) Suppose uw, vw , and wx separate uvw from triangles tuv, vwy , and wxz , where t, y, z are distinct by (2.1). If $y \in N(z)$, then wyz is a triangular face since $d(w) = 4$ and G is 3-connected, and this contradicts (2.1); thus $y \notin N(z)$. Since $d(z) \geq 3$, there is a vertex $s \in N(z) \setminus \{w, x, y\}$. Since G is 3-connected, there is a path P from s to v that does not pass through y or z . Since u, x , and w have no unknown neighbors, the first lettered vertex along P (after s) is t or v . If we choose P to have minimum length, then the three paths $wxuv, wyv$, and $wzs[P]v$ are internally disjoint. This means that $G - vw$ is 3-connected, hence a 3-polytope, and this is impossible for the reasons explained at the start of the proof of (a). \square

In the figures, we use triangles and squares to denote vertices with degree 3 and 4, respectively; all other vertices have unspecified degree. We are indebted to Enomoto

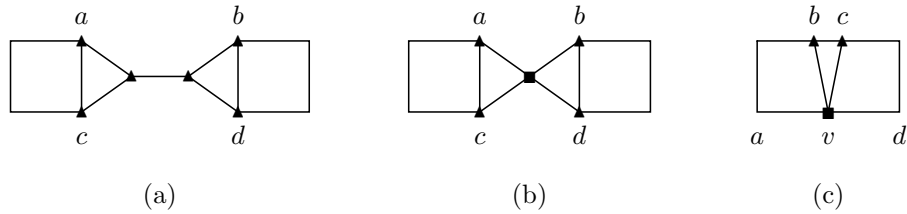


FIG. 3.1. Three reducible configurations.

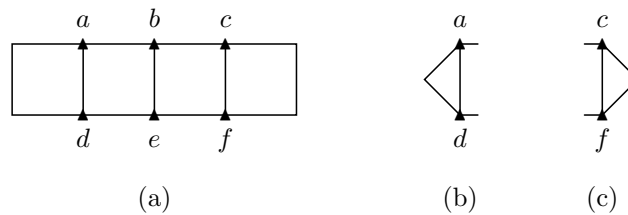


FIG. 3.2. More reducible configurations.

[7] for pointing out to us the reducibility of the configuration in Figure 3.1(c) and for showing us the proof of this below; this has significantly shortened our proof of Theorem 1.1.

LEMMA 3.2. G does not contain any of the configurations in Figure 3.1.

Proof. Suppose it does. We shall prove that $\chi_c(G) \leq k + 1$, which is a contradiction. By Lemma 2.1(a), there is an edge of G incident with b whose contraction leaves a 3-polytope H , which has a cyclic $(k + 1)$ -coloring by the minimality of G .

First suppose that G contains the configuration of Figure 3.1(a) or (b). Transfer the coloring of H to G and uncolor a, b, c, d . Since G is 3-connected, the central edge or vertex has different faces above and below it; thus a does not see d and b does not see c . Since $\Delta^*(G) \leq k$, each of these vertices currently sees at most $k - 1$ colored vertices across faces, and so there are two of the $k + 1$ colors available to give to it. Thus these vertices are easily colored (since the circuit graph C_4 is list-2-colorable). This contradicts the choice of G .

Now suppose that G contains the configuration of Figure 3.1(c). Transfer the coloring of H to G and uncolor b ; note that a and d have different colors. Each of b, c sees at most $k + 1$ vertices across faces. If b cannot be given the same color as d , then c sees two vertices with this color; so uncolor c and then color b and c in this order. This is possible since, at the time of coloring, each of these vertices sees at most k different colors. \square

LEMMA 3.3. G does not contain the configuration in Figure 3.2(a) nor any configuration obtained from it by replacing either or both of the end squares by triangles as in Figure 3.2(b) (instead of the left end square) or (c) (instead of the right end square).

Proof. In each case we prove that if G contains the configuration in question, then $\chi_c(G) \leq k + 1$, which is a contradiction.

Suppose first that G contains the configuration in Figure 3.2(a) itself. Contract

an edge and cyclically $(k + 1)$ -color the resulting 3-polytope as in the proof of Lemma 3.2; then transfer this coloring to G and uncolor the six labelled vertices. For $x \in \{a, b, c, d, e, f\}$, let $L(x)$ be the set of colors that are available to give to x , that is, that are not present on vertices that x sees. Since $\Delta^*(G) \leq k$, if $x \in \{b, e\}$, then there are at most $k - 3$ colored vertices that are seen by x and so $|L(x)| \geq 4$, whereas if $x \in \{a, c, d, f\}$, then $|L(x)| \geq 3$. Moreover, $L(a) \subseteq L(b) \supseteq L(c)$ and $L(d) \subseteq L(e) \supseteq L(f)$.

If there are distinct colors $x \in L(a) \cap L(f)$ and $y \in L(c) \cap L(d)$, then use those colors on those vertices; b and e are easily colored. If $x \in L(a) \cap L(f)$ and $(L(c) \cap L(d)) \setminus \{x\} = \emptyset$, then color a, f with x and b, c from $L(c) \setminus \{x\}$, then color e, d in that order. Finally, if $L(a) \cap L(f) = \emptyset$, then $|L(c) \setminus L(f)| \geq 2$; color b, c from $L(c) \setminus L(f)$ and color a, d, e, f in that order.

At each stage in the above process, at the moment when we color a vertex, it sees vertices of at most k different colors, and so there is a color available to give to it. If we replace either or both of the end squares by triangles as in Figure 3.2(b) and (c), then that simply increases the number of colors available for some vertices. \square

4. Completion of the proof of Theorem 1.1(a). Here $G = (V, E, F)$ is a minimal counterexample to Theorem 1.1(a). Suppose we assign a “charge” $M_0(x)$ to each element $x \in V \cup F$, where

$$M_0(x) := \begin{cases} d(x) - 6 & \text{if } x \in V, \\ 2d(x) - 6 & \text{if } x \in F. \end{cases}$$

Euler’s formula $|V| - |E| + |F| = 2$ can be rewritten in the form $(2|E| - 6|V|) + (4|E| - 6|F|) = -12$, which implies that

$$(4.1) \quad \sum_{x \in V \cup F} M_0(x) = \sum_{v \in V} (d(v) - 6) + \sum_{f \in F} (2d(f) - 6) = -12.$$

We shall now redistribute the charge, without changing its sum, in such a way that the sum is provably nonnegative, and this contradiction will prove the theorem. The rules for redistribution are as follows; let $\delta := \frac{3}{10}$.

- R0: Each face distributes its charge equally to its incident vertices.
- R1: If an edge vw separates a face f from a triangle, where $d(f) \leq 6$, $d(v) \geq 4$, and $d(w) = 3$ (so that w is a $(3, \leq 6, \geq k - 2)$ -vertex by (2.1)), then v gives w $\frac{1}{2}$ if $d(f) = 4$ and δ if $d(f) = 5$ or 6 .
- R2: If an edge uw separates a face f from a triangle uvw , where $d(f) \leq 6$ and $d(u) = d(w) = 3$ (so that u and w are $(3, \leq 6, \geq k - 2)$ -vertices), then v gives each of u and w $\frac{1}{2}$ if $d(f) = 4$ and δ if $d(f) = 5$ or 6 .
- R3: If edges vw and wx separate a 4-face uvw from two triangles, where $d(v) \geq 4$, $d(w) = 4$, and $d(x) = 3$ (so that w and x have types $(3, 4, 3, \geq k - 1)^\circ$ and $(3, 4, k)$, respectively, and w gives $\frac{1}{2}$ to x by R1), then v gives $\frac{1}{4}$ to w .
- R4: If $f = uvwx$ is a 4-face and the edges wx and xu separate f from two triangles, and $d(u) = d(w) = 3$ and $d(x) = 4$ (so that u and w are both $(3, 4, k)$ -vertices and x has type $(3, 4, 3, \geq k - 1)^\circ$, and x gives $\frac{1}{2}$ to each of u, w by R1), then v gives $\frac{2}{3}$ to x .

Note that in R1 and R2 it is not possible that $d(f) = 3$, since there are no $(3, 3, \leq k)$ -vertices by (2.1). For similar reasons, for fixed vertices v and z , v cannot be required to make a contribution to z by more than one of rules R1–R4.

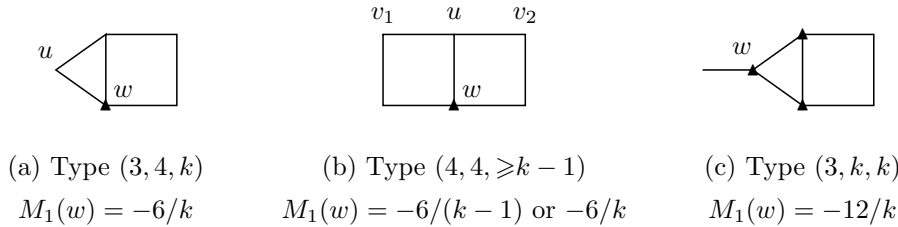


FIG. 4.1. *N-vertices.*

Let the resulting charge of each vertex v be $M_1(v)$, so that $\sum_{v \in V} M_1(v) = -12$ by (4.1). (Clearly each face now has charge 0, by R0.) It is easy to check that if w is a vertex of type $(3, 4, k)$ or $(4, 4, \geq k - 1)$, or a vertex of type $(3, k, k)$ occurring in the configuration of Figure 4.1(c), then $M_1(w)$ has the negative value given in Figure 4.1. Call such vertices *N-vertices* and all other vertices *P-vertices*. (There is no implication at this stage that if v is a P-vertex, then $M_1(v) \geq 0$, but this will eventually turn out to be the case.)

If w is the N-vertex in Figure 4.1(c), then the neighbor of w that is not shown in the figure is a P-vertex, since otherwise G would contain the configuration in Figure 3.1(a), which was shown in Lemma 3.2 to be impossible.

If w is the N-vertex in Figure 4.1(a), and if u is also an N-vertex, then u is of type $(3, k, k)$, since type $(3, 4, k)$ is impossible by the nonexistence of the configuration in Figure 3.1(c). Thus if v is the neighbor of u that is not shown in the figure, then v is a P-vertex. In this case we refer to the path vuw as an *L-path*.

If w is the N-vertex in Figure 4.1(b), and if all three neighbors of w are also N-vertices, then either v_1 or v_2 is a P-vertex, since otherwise G would contain one of the configurations that was shown to be impossible in Lemma 3.3. In this case, if v_i is a P-vertex, then we refer to the path $v_i w$ as an *L-path*.

Thus every N-vertex is either adjacent to a P-vertex or can be reached from a P-vertex by an L-path of length two whose central vertex is an N-vertex. We have a final rule for redistributing charge.

R5: Each N-vertex w receives an amount $-M_1(w)$ from a P-vertex (chosen at random) that is adjacent to w if possible and that otherwise can be reached from w by an L-path.

Let the resulting charge of each vertex v be $M_2(v)$, so that $\sum_{v \in V} M_2(v) = -12$. The following lemma provides the contradiction that completes the proof of Theorem 1.1(a).

LEMMA 4.1. $M_2(v) \geq 0$ for each vertex v .

Proof. Clearly $M_2(v) = 0$ if v is an N-vertex. So suppose v is a P-vertex. Let $\epsilon := 6/(k - 2) \leq \frac{1}{20}$, since $k \geq 122$.

If the amount that v gives to a vertex w by R5 is considered as flowing along the edge vw if there is one, or along an L-path vuw otherwise, then it is not difficult to see from Figure 4.1 that the total amount that flows out of v by R5 along an edge vu is zero unless the faces incident with edge vu have certain specified degrees, in which case the amount that flows out is bounded above by the figures in Table 4.1. To see this, first note that vu carries nothing from v by R5 if u is a P-vertex; so let us suppose that u is an N-vertex. If vu lies between two k -faces, then the other two

TABLE 4.1

Face-degrees	R5-loss <
3 and k	ϵ
3 and 4	ϵ
4 and 4	ϵ
4 and $\geq k - 1$	2ϵ
k and k	4ϵ

TABLE 4.2

Type	R0 \geq	R1 + ... + R4 \geq	$M_1(v) \geq$	R5 \geq	
(3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	$+\frac{1}{2}$	$-\epsilon$	$-M_1(v)$	N-vertex
(3, 5, k)	$\frac{4}{5} + (2 - \epsilon)$	$+\delta$	$\delta - \frac{1}{5} - \epsilon$	$-\epsilon$	
(3, 7, $k - 3$)	$\frac{8}{7} + \left(2 - \frac{6}{k-3}\right)$	0	$\frac{1}{7} - \frac{6}{k-3}$	$-\epsilon$	
(3, k , k)	$2(2 - \epsilon)$	$-2(\frac{1}{2})$	-2ϵ	$-M_1(v)$	N-vertex
(3, k , k)	$2(2 - \epsilon)$	-2δ	$1 - 2\delta - 2\epsilon$	-6ϵ	P-vertex
(4, 4, $k - 1$)	$\frac{1}{2} + \frac{1}{2} + (2 - \epsilon)$	0	$-\epsilon$	$-M_1(v)$	N-vertex
(4, 5, $k - 2$)	$\frac{1}{2} + \frac{4}{5} + (2 - \epsilon)$	0	$\frac{3}{10} - \epsilon$	-2ϵ	
(4, k , k)	$\frac{1}{2} + 2(2 - \epsilon)$	$-\frac{2}{3}$	$\frac{5}{6} - 2\epsilon$	-8ϵ	
(5, 5, $k - 3$)	$\frac{4}{5} + \frac{4}{5} + \left(2 - \frac{6}{k-3}\right)$	0	$\frac{3}{5} - \frac{6}{k-3}$	0	
(3, 3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	0	$\frac{1}{2} - \epsilon$	-3ϵ	(see
(3, 3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	$-\frac{1}{4}$	$\frac{1}{4} - \epsilon$	-2ϵ	text
(3, 3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	$-\frac{1}{2} + \frac{1}{4}$	$\frac{1}{4} - \epsilon$	-2ϵ	and
(3, 3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	$-\frac{1}{2} - \frac{1}{2} + \frac{2}{3}$	$\frac{1}{6} - \epsilon$	-2ϵ	Fig. 4.2)
(3, 3, 5, k)	$\frac{4}{5} + (2 - \epsilon)$	$-\delta - \delta$	$\frac{4}{5} - 2\delta - \epsilon$	-2ϵ	
(3, 3, k , k)	$2(2 - \epsilon)$	$-2(\frac{1}{2}) - 2\delta$	$1 - 2\delta - 2\epsilon$	-6ϵ	
(3, 4, 4, k)	$\frac{1}{2} + \frac{1}{2} + (2 - \epsilon)$	$-\frac{1}{2}$	$\frac{1}{2} - \epsilon$	-6ϵ	
(3, 4, 4, k)	$\frac{1}{2} + \frac{1}{2} + (2 - \epsilon)$	$-\frac{1}{4} - \frac{1}{4}$	$\frac{1}{2} - \epsilon$	-6ϵ	
(3, 4, 5, $k - 3$)	$\frac{1}{2} + \frac{4}{5} + \left(2 - \frac{6}{k-3}\right)$	$-\frac{1}{2} - \delta$	$\frac{4}{5} - \delta - \frac{6}{k-3}$	-2ϵ	
(3, 4, 5, k)	$\frac{1}{2} + \frac{4}{5} + (2 - \epsilon)$	$-\frac{1}{2} - \delta$	$\frac{4}{5} - \delta - \epsilon$	-3ϵ	
(3, 4, $k - 2$, k)	$\frac{1}{2} + 2(2 - \epsilon)$	-2δ	$\frac{5}{2} - 2\delta - 2\epsilon$	-5ϵ	
(3, 4, k , k)	$\frac{1}{2} + 2(2 - \epsilon)$	$-2(\frac{1}{2}) - \frac{2}{3}$	$\frac{5}{6} - 2\epsilon$	-8ϵ	
(4, 4, 4, k)	$\frac{3}{2} + (2 - \epsilon)$	0	$\frac{3}{2} - \epsilon$	-6ϵ	
(4, 4, k , k)	$\frac{2}{2} + 2(2 - \epsilon)$	$-\frac{2}{3} - \frac{2}{3}$	$\frac{5}{3} - 2\epsilon$	-9ϵ	
(4, k , k , k)	$\frac{1}{2} + 3(2 - \epsilon)$	$-\frac{2}{3}$	$\frac{23}{6} - 3\epsilon$	-12ϵ	
(k , k , k , k)	$4(2 - \epsilon)$	0	$6 - 4\epsilon$	-16ϵ	

neighbors w_1, w_2 of u are also N-vertices, and vu carries $12/k < 2\epsilon$ to u and $6/k < \epsilon$ to each of w_1, w_2 along the L-paths vuw_1 and vuw_2 . If vu is the first edge of any other L-path vuw , then vu lies between a 4-face and a $(\geq k - 1)$ -face and carries at most $6/(k - 1) < \epsilon$ to u and $6/(k - 1)$ along vuw to w . In all other cases, vu carries at most $6/(k - 1)$ from v to u .

There are now two cases to consider.

Case 1. $d(v) = 3$ or 4. Information about some types of vertices with these degrees is tabulated as Table 4.2. The reader may readily verify that the sum of the entries in the columns headed " $M_1(v) \geq$ " and " $R5 \geq$ " is nonnegative in each row of

Table 4.2 that does not describe an N-vertex. This implies that $M_2(v) \geq 0$ for vertices of these types. We must now explain some of the entries in Table 4.2 and also show that Table 4.2 covers all relevant cases.

Note that v starts with charge $d(v) - 6$ and receives charge $2 - 6/d$ from each incident d -face by R0. The amount that v receives by R0 is at least as large as the amount shown in the column headed “R0 \geq ” in Table 4.2. It is left to the reader to verify that the contribution to v by R1–R4 is bounded below by the figure in the column headed “R1+ \dots +R4 \geq .” The contributions of $+\frac{1}{2}$ and $+\delta$ in the top two rows in this column come by R1 or R2, depending on whether the edge between the 3-face and the 4-face or 5-face joins v to a (≥ 4)-vertex or to a 3-vertex, respectively. Otherwise, entries given as $-\frac{1}{2}$ and $-\delta$ come from R1, entries given as $-2(\frac{1}{2})$ and -2δ come from R2, entries of $+\frac{1}{4}$ and $-\frac{1}{4}$ come from R3, and entries of $+\frac{2}{3}$ and $-\frac{2}{3}$ come from R4.

The amount shown in the column headed “R5 \geq ” is (minus) the maximum permitted by Table 4.1, except for vertices of type $(3, 3, 4, k)$, which we shall look at more carefully below. For example, if a vertex of type $(3, 3, k, k)$ occurs as $(3, k, 3, k)^\circ$, then by Table 4.1 it can give at most 4ϵ (ϵ along each edge between a 3-face and a k -face), while if it occurs as $(3, 3, k, k)^\circ$, then it can give at most 4ϵ along the edge between the two k -faces and ϵ along each edge between a 3-face and a k -face, a total of 6ϵ . Since 6ϵ is larger, this is the figure used in Table 4.2. (This is wasteful, since the bounds given in the columns headed “R1+ \dots +R4 \geq ” and “R5 \geq ” cannot be attained simultaneously: the former is attained only by a vertex of type $(3, k, 3, k)^\circ$ and the latter only by a vertex of type $(3, 3, k, k)^\circ$. However, this waste does not affect the result, since this line of the table is no worse than that for a P-vertex of type $(3, k, k)$. In fact, the critical lines in the table are those for P-vertices of types $(3, 5, k)$ and $(3, k, k)$.) In a similar way, a vertex of type $(3, 4, 4, k)$ can give at most 5ϵ if it occurs as $(3, 4, 4, k)^\circ$ and 6ϵ if it occurs as $(4, 3, 4, k)^\circ$, and one of type $(4, 4, k, k)$ can give at most 9ϵ if it occurs as $(4, 4, k, k)^\circ$ and 8ϵ if it occurs as $(4, k, 4, k)^\circ$.

We must now show that Table 4.2 covers all relevant cases. Since $d_c(v) \geq k + 1$ for each $v \in V$ by (2.1), there are no vertices of types $(3, 4, \leq k - 1)$, $(3, 5, \leq k - 2)$, etc. The figures given in Table 4.2 for vertices of type $(3, 5, k)$ also cover vertices of types $(3, 5, k - 1)$ and $(3, 6, \geq k - 2)$; that is, the same entries in columns 2–5 of the table will work for these types. (Note that the final $-\epsilon$ in this and the next row comes from the top row of Table 4.1 and represents the loss along an edge between a 3-face and a k -face; thus it is needed only when there is a face with degree k .) The figures given for $(3, 7, k - 3)$ cover all remaining types $(3, x, y)$ with $x \leq y$ and $x \leq k - 3$. This is because, for a vertex v of given degree and cyclic degree, the sum of the charges of $2 - 6/d$ that v receives from its incident faces by R0 is smallest when the degrees of those faces are as unequal as possible; and increasing the cyclic degree cannot make matters worse, except that when $y = k$ there is the additional loss of ϵ by R5. Similarly, the figures given for $(3, k, k)$ cover all types $(3, x, y)$ with $k - 2 \leq x \leq y$. (The reason for breaking at $x = k - 2$ is that R2 requires v to be incident with two faces with degree at least $k - 2$.) The figures given for $(4, 4, k - 1)$ cover $(4, 4, k)$ as well; $(4, 5, k - 2)$ covers all remaining cases $(4, x, y)$ except for $(4, k, k)$; and $(4, k, k)$ and $(5, 5, k - 3)$ cover all other possibilities with degree 3.

There are no vertices of type $(3, 3, 3, \leq k)$ by Lemma 3.1(a). The figures given in Table 4.2 for $(3, 3, 4, k)$ cover $(3, 3, 4, k - 1)$ as well. The different possibilities are shown in Figure 4.2, in which \bullet denotes a vertex with degree *at least* 4. Here the vertices marked P can easily be seen to be P-vertices, either because they have degree

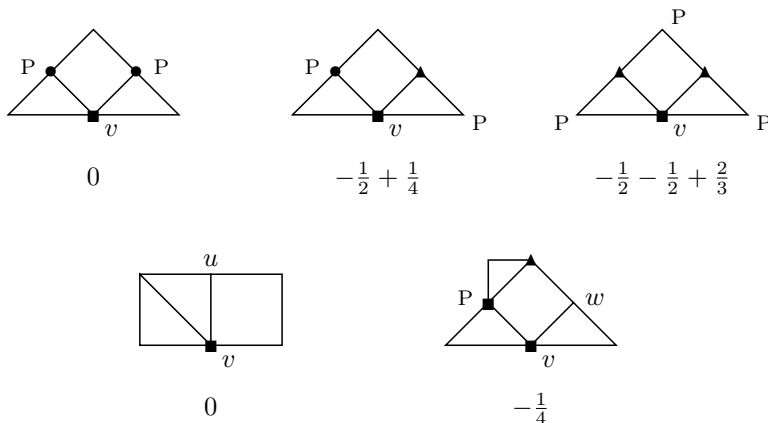


FIG. 4.2. Vertices of type $(3, 3, 4, \geq k - 1)$.

at least 4 or because they are incident with two faces of large degree; this is not one of the conditions used in distinguishing the cases. Note that $d(u) \geq 4$ and $d(w) \geq 4$ by Lemma 3.1(a) and (b), respectively; hence both are P-vertices. Nothing can be lost from v by R5 along an edge joining v to a P-vertex; using this and Table 4.1, it is easy to verify the bounds given in Table 4.2 for loss by R5.

The figures for $(3, 3, 5, k)$ in Table 4.2 cover all cases $(3, 3, x, y)$ with $x \leq y$ and $5 \leq x \leq k - 3$, and $(3, 3, k, k)$ covers all cases with $k - 2 \leq x \leq y$; note that a $(3, 3, k, k)$ -vertex cannot lose four halves by R2 because of the nonexistence of the configuration in Figure 3.1(b). Similarly, a $(3, 4, 4, k)$ -vertex cannot lose two halves by R1 because of the nonexistence of the configuration in Figure 3.1(c) (although a $(3, 4, 4, k)$ -vertex can lose two quarters by R3). Type $(3, 4, 4, k)$ covers $(3, 4, 4, x)$ (which implies $x \geq k - 2$ by (2.1)); $(3, 4, 5, k - 3)$ and $(3, 4, 5, k)$ cover all cases $(3, x, y, z)$ with $4 \leq x \leq y \leq z$ and $5 \leq y \leq k - 3$; $(3, 4, k - 2, k)$ covers all such cases with $y = k - 2$ or $k - 1$; and $(3, 4, k, k)$ covers the case $y = k$. Finally, $(4, 4, 4, k)$ to (k, k, k, k) cover all remaining cases. This completes the discussion of Case 1.

Case 2. $d(v) \geq 5$. Every charge that v gives to another vertex by R1–R3 can be thought of as being transmitted along an edge separating a triangle from a non-triangular face f . For the sole purpose of evaluating $M_1(v)$, imagine that this charge is given instead to the face f . Imagine also that if v receives $\frac{2}{3}$ across a 4-face f by R4, then v receives this $\frac{2}{3}$ from f ; and if v gives $\frac{2}{3}$ across f by R4, then v gives this $\frac{2}{3}$ equally to the two k -faces that are incident with both v and f . With this new interpretation, let $c(f, v)$ denote the net amount that f gives v by R0–R4. Then $c(f, v) \geq (2 - 6/3) - 0 = 0$ if $d(f) = 3$.

If $d(f) = 4$, say $f = uvwx$, then v cannot give more than $\frac{1}{2}$ to f by R1–R4. To see this, note that if v gives $\frac{1}{2}$ to w by R1, then v cannot also give $\frac{1}{4}$ to u by R3 in view of Lemma 3.1(b); and if v gives $\frac{1}{2}$ to u and $\frac{1}{2}$ to w by R1, then v also receives $\frac{2}{3}$ from x by R4. Thus $c(f, v) \geq (2 - 6/4) - \frac{1}{2} = 0$ if $d(f) = 4$.

If $d(f) = 5$ or 6, then v can give 2δ to f by R1 if both edges incident with v and f join v to 3-vertices and separate f from 3-faces, but v cannot give anything to f by R2–R4. If $7 \leq d(f) \leq k - 3$, then v cannot give anything at all to f . If $k - 2 \leq d(f) \leq k$, then, for each of the two edges incident with v and f , v can give f

up to $\frac{1}{2}$ by R2 or $\frac{1}{3}$ by R4; the maximum contribution would thus comprise two gifts of $\frac{1}{2}$. Thus

$$(4.2) \quad c(f, v) \geq \begin{cases} 0 & \text{if } d(f) \leq 4, \\ (2 - 6/5) - 2\delta = \frac{1}{5} & \text{if } d(f) = 5, \\ (2 - 6/6) - 2\delta = \frac{2}{5} & \text{if } d(f) = 6, \\ (2 - 6/d(f)) > 1 & \text{if } 7 \leq d(f) \leq k - 3, \\ (2 - 6/d(f)) - 1 \geq \frac{19}{20} & \text{if } d(f) \geq k - 2 \geq 120. \end{cases}$$

Thus $c(f, v) \geq 0$ always, and so $M_1(v) \geq M_0(v) = d(v) - 6 > 4\epsilon d(v)$ if $d(v) \geq 8$, since $\epsilon \leq \frac{1}{20}$. Since the loss from v by R5 is at most $4\epsilon d(v)$ by Table 4.1, this proves that $M_2(v) > 0$ if $d(v) \geq 8$.

So suppose that $5 \leq d(v) \leq 7$. Suppose v is incident with exactly t (≥ 7)-faces. Then $t \geq 1$, since if v is incident with $d(v) - 1$ (≤ 6)-faces and one further face f , then $123 \leq k + 1 \leq d_c(v) \leq (d(v) - 1)(6 - 2) + (d(f) - 2)$ so that

$$(4.3) \quad d(f) \geq 129 - 4d(v) > 100.$$

It is not difficult to see from Table 4.1 that the loss from v by R5 cannot exceed $t(4\epsilon) + (d(v) - t)\epsilon \leq \frac{d(v)+3t}{20}$. Each (≥ 7)-face incident with v gives v a net amount of at least $\frac{19}{20}$, and so $M_2(v) \geq d(v) - 6 + \frac{19}{20}t - \frac{d(v)+3t}{20} = \frac{19}{20}d(v) + \frac{4}{5}t - 6 > 0$ if $d(v) \geq 6$ or $t \geq 2$. However, if $d(v) = 5$ and $t = 1$, then v can give nothing by R2 and R4 (both of which require v to be incident with at least two ($\geq k - 2$)-faces), and so $c(f, v) = 2 - 6/d(f) > 2 - \frac{6}{100}$ by (4.3) if f is the (≥ 7)-face incident with v . Since $\frac{d(v)+3t}{20} = \frac{8}{20}$ in this case, it follows that $M_2(v) > (5 - 6) + (2 - \frac{6}{100}) - \frac{8}{20} > 0$. This completes the proof of Lemma 4.1 and hence the proof of Theorem 1.1(a). \square

5. Completion of the proof of Theorem 1.1(b). This is very similar to the proof of Theorem 1.1(a), but simpler. We shall simply indicate the differences. Define $\delta := \frac{3}{10}$ as before, but now let $\epsilon := 6/(k - 1) \leq \frac{1}{10}$, since $k \geq 61$. Now $G = (V, E, F)$ is a minimal counterexample to Theorem 1.1(b), and so $d_c(v) \geq k + 2$ for each $v \in V$ by Lemma 2.2; hence there are now no $(3, 4, k)$ -vertices or $(3, 6, k - 2)$ -vertices. The discharging rules are exactly as before, but some of them are now redundant because of the absence of $(3, 4, k)$ -vertices; thus R3 and R4 cannot arise, and v can give at most δ by R1 and at most 2δ by R2; moreover, because of the absence of $(3, \leq 6, \leq k - 2)$ -vertices, R2 applies only if v is incident with at least two ($\geq k - 1$)-faces. Because of the absence of $(3, 4, k)$ -vertices, two of the three types of N-vertex in Figure 4.1 can no longer exist; the only N-vertices are of type $(4, 4, k)$ for which $M_1(v) = -6/k > -\epsilon$. Thus the amount that v gives by R5 is at most

$$(5.1) \quad \begin{array}{l} 2\epsilon \text{ along an edge between a 4-face and a } k\text{-face,} \\ \epsilon \text{ along an edge between two 4-faces,} \\ 0 \text{ along all other edges.} \end{array}$$

As before, $\sum_{v \in V} M_2(v) = -12$, and we obtain a contradiction by proving the following lemma.

LEMMA 5.1. $M_2(v) \geq 0$ for each vertex v .

Proof. Clearly $M_2(v) = 0$ if v is an N-vertex. So suppose v is a P-vertex.

Case 1. $d(v) = 3$ or 4. Here we use Table 5.1 in exactly the same way as we used Table 4.2 previously.

TABLE 5.1

Type	R0 \geq	R1 + ... + R4 \geq	$M_1(v) \geq$	R5 \geq
(3, 5, k)	$\frac{4}{5} + (2 - \epsilon)$	$+\delta$	$\delta - \frac{1}{5} - \epsilon$	0
(3, 7, $k - 2$)	$\frac{8}{7} + \left(2 - \frac{6}{k-2}\right)$	0	$\frac{1}{7} - \frac{6}{k-2}$	0
(3, k , k)	$2(2 - \epsilon)$	-2δ	$1 - 2\delta - 2\epsilon$	0
(4, 4, k)	$\frac{1}{2} + \frac{1}{2} + (2 - \epsilon)$	0	$-\epsilon$	$-M_1(v)$ N-vertex
(4, 5, k)	$\frac{1}{2} + \frac{4}{5} + (2 - \epsilon)$	0	$\frac{3}{10} - \epsilon$	-2ϵ
(3, 3, 4, k)	$\frac{1}{2} + (2 - \epsilon)$	0	$\frac{1}{2} - \epsilon$	-2ϵ
(3, 3, 5, k)	$\frac{4}{5} + (2 - \epsilon)$	$-\delta - \delta$	$\frac{4}{5} - 2\delta - \epsilon$	0
(3, 3, k , k)	$2(2 - \epsilon)$	$-2\delta - 2\delta$	$2 - 4\delta - 2\epsilon$	0
(3, 4, 4, k)	$\frac{1}{2} + \frac{1}{2} + (2 - \epsilon)$	0	$1 - \epsilon$	-4ϵ
(3, 4, 5, k)	$\frac{1}{2} + \frac{4}{5} + (2 - \epsilon)$	$-\delta$	$\frac{13}{10} - \delta - \epsilon$	-2ϵ
(3, 4, k , k)	$\frac{1}{2} + 2(2 - \epsilon)$	-2δ	$\frac{5}{2} - 2\delta - 2\epsilon$	-4ϵ
(4, 4, 4, $k - 2$)	$\frac{3}{2} + \left(2 - \frac{6}{k-2}\right)$	0	$\frac{3}{2} - \frac{6}{k-2}$	-2ϵ
(4, 4, 4, k)	$\frac{3}{2} + (2 - \epsilon)$	0	$\frac{3}{2} - \epsilon$	-6ϵ

There are now no (3, 4, k)-vertices. The figures given in Table 5.1 for (3, 5, k) cover (3, 6, $\geq k - 1$) as well; (3, 7, $k - 2$) covers all remaining cases (3, x , y) with $x \leq y$ and $x \leq k - 2$, and (3, k , k) covers all cases with $k - 1 \leq x \leq y$. (The reason for breaking at $x = k - 1$ is that R2 now requires v to be incident with two faces with degree at least $k - 1$.) The next type, (4, 4, k), is the N-type, and (4, 5, k) covers all other possibilities with degree 3.

As before, there are no vertices of type (3, 3, 3, $\leq k$), and now there are none of type (3, 3, 4, $\leq k - 1$) either. The figures given in Table 5.1 for (3, 3, 5, k) cover all cases (3, 3, x , y) with $5 \leq x \leq y$ and $x \leq k - 2$, and (3, 3, k , k) covers those with $x \geq k - 1$; (3, 4, 4, k) covers (3, 4, 4, $k - 1$); (3, 4, 5, k) covers all cases (3, 4, x , y) with $5 \leq x \leq y$ and $x \leq k - 2$, and (3, 4, k , k) covers those with $x \geq k - 1$. Finally, the figures given for (4, 4, 4, $k - 2$) and (4, 4, 4, k) cover all remaining cases. This completes the discussion of Case 1.

Case 2. $d(v) \geq 5$. With the same conventions as in the proof of Lemma 4.1, for each face f incident with v , the net amount $c(f, v)$ that f gives v satisfies

$$(5.2) \quad c(f, v) \geq \begin{cases} (2 - 6/3) - 0 = 0 & \text{if } d(f) = 3, \\ (2 - 6/4) - 0 = \frac{1}{2} & \text{if } d(f) = 4, \\ (2 - 6/5) - 2\delta = \frac{1}{5} & \text{if } d(f) = 5, \\ (2 - 6/6) - 2\delta = \frac{2}{5} & \text{if } d(f) = 6, \\ (2 - 6/d(f)) > 1 & \text{if } 7 \leq d(f) \leq k - 2, \\ (2 - 6/d(f)) - 2\delta > 1 & \text{if } d(f) \geq k - 1 \geq 60. \end{cases}$$

(Apart from the different lower bound for k , the only differences in (5.2) from (4.2) are attributable to the nonexistence of (3, ≤ 6 , $\leq k - 2$)-vertices, which means that v gives nothing to f by R2 unless $d(f) \geq k - 1$, and the nonexistence of (3, 4, k)-vertices, which means that v gives at most 2δ to f by R2 if $d(f) \geq k - 1$ and nothing to f by R1 if $d(f) = 4$.) Thus $c(f, v) \geq 0$ always, and $M_1(v) \geq M_0(v) = d(v) - 6 > 2\epsilon d(v)$ if $d(v) \geq 8$, since $\epsilon \leq \frac{1}{10}$. Since the loss from v by R5 is at most $2\epsilon d(v)$ by (5.1), this proves that $M_2(v) > 0$ if $d(v) \geq 8$.

So suppose that $5 \leq d(v) \leq 7$. Suppose v is incident with exactly t (≥ 7)-faces.

Then $t \geq 1$, since if v is incident with $d(v) - 1$ (≤ 6)-faces and one further face f , then $62 \leq k + 1 \leq d_c(v) \leq (d(v) - 1)(6 - 2) + (d(f) - 2)$ so that

$$(5.3) \quad d(f) \geq 68 - 4d(v) \geq 40.$$

By (5.1), the loss from v by R5 cannot exceed $2t(2\epsilon) + (d(v) - 2t)\epsilon \leq \frac{d(v)+2t}{10}$. Each (≥ 7)-face incident with v gives v a net amount of more than 1, and so $M_2(v) > d(v) - 6 + t - \frac{d(v)+2t}{10} = \frac{9}{10}d(v) + \frac{8}{10}t - 6 > 0$ if $d(v) \geq 6$ or $t \geq 2$. However, if $d(v) = 5$ and $t = 1$, then v can give nothing by R2 (which requires v to be incident with at least two ($\geq k - 1$)-faces), and so $c(f, v) = 2 - 6/d(f) \geq 2 - \frac{6}{40}$ by (5.3) if f is the (≥ 7)-face incident with v . Since $\frac{d(v)+2t}{10} = \frac{7}{10}$ in this case, it follows that $M_2(v) \geq (5 - 6) + (2 - \frac{6}{40}) - \frac{7}{10} > 0$. This completes the proof of Lemma 5.1 and hence the proof of Theorem 1.1(b). \square

REFERENCES

- [1] K. ANDO, H. ENOMOTO, AND A. SAITO, *Contractible edges in 3-connected graphs*, J. Combin. Theory Ser. B, 42 (1987), pp. 87–93.
- [2] O. V. BORODIN, *Solution of Ringel's problem on vertex-face coloring of plane graphs and coloring of 1-planar graphs*, Metody Diskret. Analiz., 41 (1984), pp. 12–26 (in Russian).
- [3] O. V. BORODIN, *A new proof of the 6 color theorem*, J. Graph Theory, 19 (1995), pp. 507–521.
- [4] O. V. BORODIN, *Cyclic degree and cyclic coloring of 3-polytopes*, J. Graph Theory, 23 (1996), pp. 225–231.
- [5] O. V. BORODIN, D. P. SANDERS, AND Y. ZHAO, *On cyclic colorings and their generalizations*, Discrete Math., 203 (1999), pp. 23–40.
- [6] O. V. BORODIN AND D. R. WOODALL, *Cyclic degrees of 3-polytopes*, Graphs Combin., 15 (1999), pp. 267–277.
- [7] H. ENOMOTO, *private communication*, 1997.
- [8] H. ENOMOTO, M. HORŇÁK, AND S. JENDROL', *Cyclic chromatic number of 3-connected plane graphs*, SIAM J. Discrete Math., 14 (2001), pp. 121–137.
- [9] R. HALIN, *Zur Theorie der n-fach zusammenhängenden Graphen*, Abh. Math. Sem. Univ. Hamburg, 33 (1969), pp. 133–164.
- [10] M. HORŇÁK AND S. JENDROL', *On vertex types and cyclic colourings of 3-connected plane graphs*, Discrete Math., 212 (2000), pp. 101–109.
- [11] O. ORE AND M. D. PLUMMER, *Cyclic coloration of plane graphs*, in Recent Progress in Combinatorics, W. T. Tutte, ed., Academic Press, New York, 1969, pp. 287–293.
- [12] M. D. PLUMMER AND B. TOFT, *Cyclic coloration of 3-polytopes*, J. Graph Theory, 11 (1987), pp. 507–515.
- [13] G. RINGEL, *Ein Sechsfarbenproblem auf der Kugel*, Abh. Math. Sem. Univ. Hamburg, 29 (1965), pp. 107–117.
- [14] E. STEINITZ, *Polyheder und Raumeinteilungen*, Enzykl. math. Wiss., 3 (Geometrie), Part 3AB 12 (1922), pp. 1–139.

ON SOME POLYNOMIALS RELATED TO WEIGHT ENUMERATORS OF LINEAR CODES*

ALEXANDER BARG[†]

Abstract. A linear code can be thought of as a vector matroid represented by the columns of the code's generator matrix; a well-known result in this context is Greene's theorem on a connection of the weight polynomial of the code and the Tutte polynomial of the matroid. We examine this connection from the coding-theoretic viewpoint, building upon the rank polynomial of the code. This enables us to obtain bounds on all-terminal reliability of linear matroids and new proofs of two known results: Greene's theorem and a connection between the weight polynomial and the partition polynomial of the Potts model.

Key words. all-terminal reliability, Greene's theorem, linear code, linear matroid, Potts model, rank polynomial, Tutte polynomial, weight enumerator

AMS subject classifications. 94B05, 05B35

PII. S0895480199364148

1. Introduction. A linear matroid M together with a chosen representation over a finite field \mathbb{F}_q is the same object as a linear code. The most well-known result underlining this connection is Greene's theorem on the relation of the weight polynomial of the code and the Tutte polynomial of the matroid. In this paper we further examine the relation between the polynomial invariants of codes, matroids, and some other combinatorial objects. Our point of view is coding-theoretic. We begin with listing basic definitions for linear codes and some very simple linear-algebraic properties of subcodes. These properties lead almost immediately to a relation between the weight polynomial of a linear code and the rank polynomial of the corresponding matroid. This relation is equivalent to Greene's theorem which is shown to be a purely linear-algebraic fact. An advantage of the coding-theoretic point of view is determined by the fact that the weight polynomial enjoys more structural properties than more general matroid invariants; when this structure translates to other problems, it sometimes produces interesting insights.

As an example, we relate the reliability polynomial of a linear matroid to an evaluation of the weight polynomial of the code. The corresponding functional on linear codes turns out to be well studied under the name of the probability of undetected error of the code. Together with some related ideas this enables us to derive upper and lower bounds on the matroid reliability. As another application of the weight-rank connection, we give a direct proof of the link between the partition function of the Potts model and the weight polynomial of the cocycle code of the graph.

General sources for coding theory are the books [17], [19]. Relevant applications of the Tutte polynomial are covered in [6], [24]. All the necessary information on interaction models is contained in [24].

2. The rank polynomial of the code.

*Received by the editors November 8, 1999; accepted for publication (in revised form) January 2, 2002; published electronically February 27, 2002.

<http://www.siam.org/journals/sidma/15-2/36414.html>

[†]Bell Labs, Lucent Technologies, 600 Mountain Avenue, Room 2C-375, Murray Hill, NJ 07974 (abarg@research.bell-labs.com).

2.1. Definitions. A *linear code* C of length n is a linear subspace of \mathbb{F}_q^n . Let \mathcal{A}_i be the number of vectors of Hamming weight i in it, $0 \leq i \leq n$. Clearly, $\mathcal{A}_0 = 1$. The minimal $i \geq 1$ such that $\mathcal{A}_i \neq 0$ is called the *minimum distance* of the code, denoted $d(C)$. The polynomial

$$\mathcal{A}(x, y) = \sum_{i=0}^n \mathcal{A}_i x^{n-i} y^i$$

is called the *weight polynomial* of C . The matrix \mathbf{G} whose rows form a basis of C as an \mathbb{F}_q -linear space is called a *generator matrix* of the code.

Let $E = \{1, 2, \dots, n\}$ be the set of code coordinates. For any subset $F \subset E$ denote by $\mathbf{G}(F)$ the submatrix of \mathbf{G} formed by columns with numbers in F . Let $\bar{F} = E \setminus F$.

Let $Z \subset E$ be the set of all-zero columns in G . The number $n - |Z|$ is called the *effective length* of C , denoted $\text{el}(C)$.

By $(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n a_i b_i$ we denote the standard dot product in \mathbb{F}_q^n . The *dual code* of C is defined as $C^\perp := \{\mathbf{c} \in \mathbb{F}_q^n \mid (\mathbf{c}, \mathbf{c}') = 0 \text{ for all } \mathbf{c}' \in C\}$. Denote by \mathbf{H} the generator matrix of C^\perp . (This matrix is also called the *parity-check matrix* of C .) Let $k := \dim C$, so $\dim C^\perp = n - k$. The weight polynomial of C^\perp is denoted by $\mathcal{A}^\perp(x, y)$. The minimum distance of C^\perp is also called the *dual distance* of C .

Let $C_F := \text{proj}_F C$, $C^{\bar{F}} := \{\mathbf{c} \in C \mid c_e = 0 \text{ for all } e \in \bar{F}\}$. In coding literature the subcode $C^{\bar{F}}$ is called the *shortening* of C and the subcode C_F the *puncturing* of C , both with respect to \bar{F} . Clearly, $\dim C_F = \text{rk}(\mathbf{G}(F))$. Standard properties of these subcodes are given in the following obvious lemma.

LEMMA 2.1.

- (i) $C_F \cong C/C^{\bar{F}}$; $\dim C_F = k - \dim C^{\bar{F}}$,
- (ii) $\dim C^{\bar{F}} = |F| - \text{rk}(\mathbf{H}(F))$.

The *rank polynomial* of C is defined as $\mathcal{U}(x, y) = \sum_{u=0}^n \sum_{v=0}^k \mathcal{U}_u^v x^u y^v$, where

$$\mathcal{U}_u^v = |\{F \subseteq E \mid |F| = u, \text{rk}(\mathbf{G}(F)) = v\}|.$$

The code C can be also thought of as a (vector) matroid M represented by the column space of \mathbf{G} ; so given C we speak of a matroid of the code, denoted $M(C)$, and vice versa; given M , we call C the code of M , denoted $C(M)$. If $\text{el}(C) = n$, then $M(C)$ is loopless. The interest for us in pursuing this connection, besides establishing new links between linear codes and combinatorics, is that methods of coding theory enable one to derive absolute bounds on evaluations of $\mathcal{A}(x, y)$ which can be useful in other areas.

2.2. Greene's theorem. The rank polynomial of C , essentially, is an invariant of $M(C)$. Another matroid invariant that appears in numerous contexts in combinatorics is the *Tutte polynomial* of M , defined as follows:

$$\mathcal{T}(M; x, y) = \sum_{u=0}^n \sum_{v=0}^k \mathcal{U}_u^v (x-1)^{k-v} (y-1)^{u-v},$$

where $k = \dim C$ is the rank of M . The following theorem relates $\mathcal{A}(x, y)$ and $\mathcal{T}(M; x, y)$.

THEOREM 2.2 (see [8]).

$$(2.1) \quad \mathcal{A}(x, y) = y^{n-k} (x-y)^k \mathcal{T}\left(M; \frac{x+(q-1)y}{x-y}, y\right).$$

The proof, as given in [8] and reproduced in [23], [6], first shows that a certain polynomial related to $\mathcal{A}(x, y)$ is a (Tutte–Grothendieck) invariant of M , and then invokes Brylawski’s theorem that states that every invariant is an evaluation of the Tutte polynomial, defined completely by its values on loops and isthmuses. We shall show that this theorem follows from Lemma 2.1.

The polynomials $\mathcal{A}(x, y)$ and $\mathcal{U}(x, y)$ are connected by the following relation.

THEOREM 2.3.

$$(2.2) \quad \mathcal{A}(x, y) = y^n |C| \mathcal{U} \left(\frac{x-y}{y}, \frac{1}{q} \right).$$

Proof. Let us count in two ways the size of the set

$$\left\{ (F, \mathbf{c}) \mid F \subseteq E, |F| = w \text{ and } \mathbf{c} \in C^F, 0 \leq \text{wt}(\mathbf{c}) \leq w \right\}.$$

Taking into account Lemma 2.1, we obtain

$$(2.3) \quad \sum_{i=0}^w \binom{n-i}{n-w} \mathcal{A}_i = \sum_{|F|=w} |C^F| = \sum_{|F|=w} q^{k-\text{rk}(\mathbf{G}(\bar{F}))}$$

$$(2.4) \quad = \sum_{u=0}^k q^{k-u} \mathcal{U}_{n-w}^u \quad (0 \leq w \leq n).$$

Now let $\mathcal{B}_w = \sum_{j=0}^w \binom{n-j}{n-w} \mathcal{A}_j$. We then have

$$(2.5) \quad \begin{aligned} \mathcal{A}(x, y) &= \sum_{w=0}^n \mathcal{B}_w (x-y)^{n-w} y^w = \sum_{w=0}^n \sum_{u=0}^k q^{k-u} \mathcal{U}_{n-w}^u (x-y)^{n-w} y^w \\ &= \sum_{\alpha=0}^n \sum_{u=0}^k q^{k-u} \mathcal{U}_{\alpha}^u (x-y)^{\alpha} y^{n-\alpha} = y^n q^k \mathcal{U} \left(\frac{x-y}{y}, \frac{1}{q} \right). \quad \square \end{aligned}$$

Note that Theorem 2.3 already relates $\mathcal{A}(x, y)$ to a polynomial with coefficients \mathcal{U}_u^v . Therefore, Theorem 2.2 should be a mere reformulation of (2.2), which it is.

Proof of Theorem 2.2. Starting with the definition of \mathcal{T} , we obtain

$$\begin{aligned} y^{n-k} (x-y)^k \mathcal{T} \left(M; \frac{x+(q-1)y}{x-y}, \frac{x}{y} \right) \\ &= y^{n-k} (x-y)^k \sum_{u=0}^n \sum_{v=0}^k \mathcal{U}_u^v \left(\frac{qy}{x-y} \right)^{k-v} \left(\frac{x-y}{y} \right)^{u-v} \\ &= y^n q^k \mathcal{U} \left(\frac{x-y}{y}, \frac{1}{q} \right). \quad \square \end{aligned}$$

Equation (2.3) together with Lemma 2.1(ii) also enables us to relate the weight polynomial of C and the rank polynomial of C^\perp , denoted $\mathcal{U}^\perp(x, y)$.

THEOREM 2.4 (see [4]).

$$(2.6) \quad \mathcal{A}(x, y) = (x-y)^n \mathcal{U}^\perp \left(\frac{qy}{x-y}, \frac{1}{q} \right),$$

$$(2.7) \quad \mathcal{U}^\perp(x, y) = x^n y^{\dim C^\perp} \mathcal{U} \left(\frac{1}{xy}, y \right).$$

The only known application of Theorem 2.2 in coding theory [8], [23], [6] is to derive MacWilliams-type theorems on the relation of \mathcal{A} and \mathcal{A}^\perp . The classical MacWilliams equation has the form

$$(2.8) \quad \mathcal{A}^\perp(x, y) = \frac{1}{|C|} \mathcal{A}(x + (q - 1)y, x - y).$$

This was proved in [8] by using (2.1) together with the relation $T(M(C); x, y) = T(M(C^\perp); y, x)$, which is implied by the definition of the Tutte polynomial. We have shown that this argument is the same as one of the two proofs in the original paper [16].

2.3. Support weight distributions. Let us also mention a generalization of Theorem 2.2 observed in [3]. It is related to the notion of support weight distributions of linear codes.

The *support* of a subset $A \subset C$ is defined as $\text{supp } A = \bigcup_{\mathbf{c} \in A} \text{supp}(\mathbf{c})$, where $\text{supp}(\mathbf{c}) = \{e \in \{1, 2, \dots, n\} : \mathbf{c}_e \neq 0\}$.

Definition. The r th support weight distribution of a code C is the set of n numbers \mathcal{A}_i^r , $0 \leq i \leq n$, where

$$\mathcal{A}_i^r = \left| \left\{ A : A \text{ a linear subcode of } C, \dim A = r, |\text{supp } A| = i \right\} \right|.$$

In particular, for $r = 1$ we obtain the ‘‘support distribution’’ of the projective code \mathbf{PC} . So $\mathcal{A}_i = \mathcal{A}_i^0 + (q - 1)\mathcal{A}_i^1$, $0 \leq i \leq n$. The following theorem relates the support weight distributions of C to its Tutte polynomial.

THEOREM 2.5 (see [3]).

$$\sum_{i=0}^n \left(\sum_{m=0}^r [r]_m \mathcal{A}_i^m \right) x^{n-i} y^i = (x - y)^k y^{n-k} \mathcal{T} \left(M, \frac{x + (q^r - 1)y}{x - y}, \frac{x}{y} \right),$$

where $k = \dim C$ and $[r]_m := \prod_{j=0}^{m-1} (q^r - q^j)$.

The proof method of [3] parallels that of [8]. Without going into details we remark that it is possible to give a proof of Theorem 2.5 similar to that of the previous section. The proof is based on the following generalization of Lemma 2.4.

LEMMA 2.6 (see [21]).

$$\sum_{i=0}^w \binom{n-i}{n-w} \mathcal{A}_i^r = \sum_{v=0}^{n-k} \begin{bmatrix} w-v \\ r \end{bmatrix} (\mathcal{U}^\perp)_w^v \quad (0 \leq w \leq n, 0 \leq r \leq k).$$

Another proof of Theorem 2.5 is given in [20].

3. The reliability polynomial of linear matroids.

3.1. Definitions. Let M be a linear matroid of rank k on the ground set E of size n defined by its representation over \mathbb{F}_q and let $f_i := \mathcal{U}_i^i$ be its number of independent sets of size i . The (*all-terminal*) *reliability polynomial* of M , by definition, is

$$(3.1) \quad \mathcal{R}(M; x, y) := \sum_{i=0}^k f_i x^{n-i} y^i.$$

The terminology is motivated by the special case of cographic matroids. Namely, let $G(V, E)$ be a connected graph and let M be a matroid whose independent sets are

given by subsets of edges whose removal does not make G disconnected. The rank k of M equals $|E| - |V| + 1$. Further, suppose that G is subjected to an edge removal process under which each edge in E is independently left intact with probability p and removed with probability $1 - p$. Then the probability that upon completion of this process the graph remains connected is given by $\mathcal{R}(M; p, 1 - p)$. If G is thought of as a network in which each link is operational with probability p , then $\mathcal{R}(M; p, 1 - p)$ gives the probability for G to be operational. Below we use the notation $\text{Rel}(M, p) := \mathcal{R}(M; p, 1 - p)$.

One of the main problems related to the reliability polynomial in the general case is deriving bounds on $\text{Rel}(M, p)$ in terms of other numerical parameters of M . The aim of this section is to use results from coding theory to derive bounds on $\text{Rel}(M, p)$.

3.2. Upper bounds. Let $(\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_n)$ be the weight distribution of a linear k -dimensional code $C(M)$. The reliability $\text{Rel}(M, p)$ is related to the weight polynomial $\mathcal{A}(\cdot, \cdot)$ of $C(M)$, via the following inequalities.

THEOREM 3.1.

$$(3.2) \quad \text{Rel}(M, p) \leq \sum_{w=n-k+1}^n p^w (1-p)^{n-w} \sum_{j=1}^w \binom{n-j}{n-w} \mathcal{A}_j + f_k p^{n-k} (1-p)^k,$$

$$(3.3) \quad \text{Rel}(M, p) \leq \mathcal{A}(1, p) - 1 + f_k p^{n-k} (1-p)^k.$$

Proof. We have

$$\begin{aligned} \text{Rel}(M, p) - f_k p^{n-k} (1-p)^k &= \sum_{i=0}^{k-1} \mathcal{U}_i^i p^{n-i} (1-p)^i \\ &\leq \sum_{i=0}^{k-1} p^{n-i} (1-p)^i \sum_{u=0}^i (q^{k-u} - 1) \mathcal{U}_i^u \\ &= \sum_{w=n-k+1}^n p^w (1-p)^{n-w} \sum_{u=0}^k (q^{k-u} - 1) \mathcal{U}_{n-w}^u. \end{aligned}$$

Now proceeding as in (2.3), (2.5) we see that

$$\mathcal{C}_w := \sum_{u=0}^k (q^{k-u} - 1) \mathcal{U}_{n-w}^u = \sum_{j=1}^w \binom{n-j}{n-w} \mathcal{A}_j.$$

Substituting this proves (3.2). To prove (3.3), we extend the summation on w on the right-hand side of (3.2) to the range $1 \leq w \leq n$ and note that

$$\sum_{i=1}^n \mathcal{A}_i x^{n-i} y^i = \sum_{w=1}^n \mathcal{C}_w (x-y)^{n-w} y^w.$$

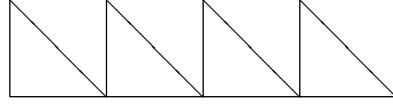
Thus

$$\sum_{w=1}^n p^w (1-p)^{n-w} \sum_{j=1}^w \binom{n-j}{n-w} \mathcal{A}_j = \sum_{i=1}^n \mathcal{A}_i p^i.$$

This completes the proof. \square

In general, bounds (3.2)–(3.3) are good only for small values of p . We give one simple example.

Example. Consider the “ladder” graph Γ from [7].



Its cocycle matroid M has rank 8 and can be represented over \mathbb{F}_2 by the columns of the matrix \mathbf{G} whose rows are $(1^3 0^{14})$, $(1^2 0^1 2^0 1^2)$, and their three right shifts by four positions. The code C generated by \mathbf{G} has parameters $[n = 17, k = 8, d = 3]$ and weight distribution $\mathcal{A}_0 = 1, \mathcal{A}_3 = 8, \mathcal{A}_4 = 7, \mathcal{A}_5 = 6, \dots$. Its dual code C^\perp (the cycle code of Γ) has parameters $[17], [9], [2]$. We have [7] $f_0 = 1, f_1 = 17, f_2 = 134, f_3 = 641, f_4 = 2041, f_5 = 4447, \dots$. On the other hand, estimate (3.2) gives $f_0 = 1, f_1 \leq 17, f_2 \leq 136, f_3 \leq 688, f_4 \leq 2499, f_5 \leq 7013, \dots$. For small p this results in reasonably good estimates of $\text{Rel}(M, p)$. We note that the well-known Ball–Provan bounds give in this case better results: $f_0 = 1, f_1 \leq 17, f_2 \leq 134, f_3 \leq 651, f_4 \leq 2184, f_5 \leq 5369, \dots$ and hence better estimates of $\text{Rel}(M, p)$.

An advantage of the estimate (3.2) is that the weight coefficients of any linear code satisfy a set of Delsarte inequalities, i.e., linear inequalities of the form

$$\sum_{u=0}^n (-1)^{j-u} \binom{n-u}{n-j} q^u \sum_{i=0}^{n-u} \binom{n-i}{u} A_i \geq 0 \quad (0 \leq j \leq n).$$

This enables one to upper bound the right-hand side of (3.2) using the methods of [1], [2]. Note that absolute bounds on the reliability $\text{Rel}(M(C), p)$ that are obtainable under this approach involve the minimum distance of the code C as a parameter.

Another way to bound above the right-hand side of (3.3) is by estimating evaluations of $\mathcal{A}(x, y)$. For them, let us look at the problem of error detection in coding theory. More specifically, given a linear code C , its *probability of undetected error* is

$$P_{ue}(C, \epsilon) := \sum_{i=1}^n \mathcal{A}_i \left(\frac{\epsilon}{q-1} \right)^i (1-\epsilon)^{n-i} = \mathcal{A} \left(1 - \epsilon, \frac{\epsilon}{q-1} \right) - (1-\epsilon)^n.$$

The motivation for this definition is the following scenario of information transmission. Suppose a q -ary code C is used to send messages over the q -ary symmetric channel. The channel is memoryless, and if a is a q -ary letter on the input, then the probability of getting a letter b on the output is given by

$$P(b|a) = \frac{\epsilon}{q-1} (1 - \delta_{a,b}) + (1 - \epsilon) \delta_{a,b}$$

for some fixed $\epsilon \in [0, (q-1)/q]$. Suppose that at the receiving end the code is used for error detection. Namely, the received vector \mathbf{y} is tested for containment in C , and, if the test fails, the decoder “detects an error.” The probability that the error will be missed (not detected) is then given by $P_{ue}(C, \epsilon)$.

Therefore, we can formulate the following proposition.

PROPOSITION 3.2.

$$(3.4) \quad \text{Rel}(M, p) \leq (1 + p(q-1))^n P_{ue} \left(C(M), \frac{p(q-1)}{1 + p(q-1)} \right) + f_k p^{n-k} (1-p)^k.$$

Proof. By Theorem 3.1 we have

$$P_{ue}(C(M), \epsilon) = (1 - \epsilon)^n \sum_{i=1}^n \mathcal{A}_i p^i \geq (1 - \epsilon)^n [\text{Rel}(M, p) - f_k p^{n-k} (1 - p)^k],$$

where $p = \epsilon / (q - 1)(1 - \epsilon)$. \square

In the context of information transmission one assumes that $\epsilon \in [0, (q - 1)/q]$ since for greater ϵ the probability of undetected error is usually close to 0. Then p varies in the entire segment $[0, 1]$; so inequality (3.4) covers all the interval of values of p .

Among the problems that present interest in coding theory are the behavior of $P_{ue}(C, p)$ for a given code (for instance, the question whether $P_{ue}(C, p)$ is monotone in p , and, if not, what is the number of its maxima), and absolute bounds on P_{ue} . More specifically, let

$$P_{ue}(n, R, p) = \min_{C \in \mathbb{F}_q^n, |C|=q^{nR}} P_{ue}(C, p)$$

be the smallest possible probability of undetected error over linear codes of fixed length n and size q^{nR} . A number of lower and upper bounds on $P_{ue}(n, R, p)$ are known in the literature; see [12]. Together with Proposition 3.2 and the obvious $U_k^k \leq \binom{n}{k}$ this enables us to formulate bounds on the reliability polynomial. For simplicity let us put $q = 2$. Let

$$\text{Rel}(n, k, p) = \min_{\substack{M \text{ is a linear matroid on } E \\ |E|=n, \text{rk } E=k}} \text{Rel}(M, p).$$

PROPOSITION 3.3.

$$(3.5) \quad \text{Rel}(n, k, p) \leq 2^{k-n} ((1 + p)^n - 1) + \binom{n}{k} p^{n-k} (1 - p)^k,$$

$$(3.6) \quad \text{Rel}(n, k, p) \leq (1 + p)^n \left(\frac{2^k - 1}{2^n - 1} \left[\frac{(p^u + (1 + p)^u)^n}{(1 + p)^{nu}} - (1 + p)^{-un} \right] \right)^{\frac{1}{u}} + \binom{n}{k} p^{n-k} (1 - p)^k \quad (0 < u \leq 1).$$

Proof. The proof follows upon substituting in (3.4) known bounds on P_{ue} , those of [13] and [14], respectively. \square

Note that (3.5) is the special case of (3.6) for $u = 1$. Although the quantity $\mathcal{A}(1, p)$ includes many more (nonnegative) terms than $\text{Rel}(M, p)$, the estimates of the last proposition are nontrivial for some values of the rank k and of p . To see this, let $n \rightarrow \infty, k = Rn, 0 < R < 1$. Let us rewrite (3.5) as follows:

$$\text{Rel}(n, nR, p) \lesssim 2^{-n \min[1 - R - \log_2(1 + p), D(R|1 - p)]},$$

where

$$D(x||y) = x \log_2(x/y) + (1-x) \log_2(1-x)/(1-y).$$

Thus for large n the estimate (3.5) is nontrivial if $p < 2^{1-R} - 1$, and roughly the same holds true for (3.6).

Because of the connection to linear codes the problem of bounding $P_{ue}(n, R, p)$ generally seems easier than that of bounding $\text{Rel}(n, k, p)$. However, the exact asymptotic behavior of $P_{ue}(n, R, p)$ is also not known, let alone the exact value of P_{ue} for finite n, k .

3.3. Lower bounds. Results from coding theory can also be used to derive lower bounds on $\text{Rel}(M, p)$. Let us quote a result from [10].

PROPOSITION 3.4. *Let M be a binary matroid of rank k on the ground set of size n and suppose that the distance of the code $C(M)$ is d . Then $f_k \geq 2^{\dim(k, d)}$, where $\dim(k, d)$ is the minimum dimension of a binary linear code of length k and dual distance d .*

Together with known bounds on the minimal dimension of linear codes of given length and dual distance [15] this gives lower bounds on f_k and hence also on $\text{Rel}(M, p)$ since $\text{Rel}(M, p) \geq f_k p^{n-k} (1-p)^k$.

Note that it is easy to understand the average behavior of the coefficients f_i if the code is chosen randomly with uniform probability from the ensemble of linear codes. This amounts to a study of coranks of submatrices of a random matrix, which is a fairly standard subject in the study of linear codes; see, e.g., [10]. A similar technique was used in the study of the average, over the ensemble of linear codes of given length and dimension, probability of undetected error [12, sect. 3.2].

4. The partition function of the Potts model. Let $\Gamma = (V, E)$ be a finite graph with $|E(\Gamma)| = n$ edges and $c(E)$ connected components. Consider the Potts model of interaction for a physical system represented by Γ [5], [24]. Under this model each vertex in $V(\Gamma)$ can be in one of q possible states; an allocation of states to all the vertices defines a state σ of the system or a coloring of $V(\Gamma)$ with q colors. Two adjacent vertices interact with nonzero energy when they have the same color; the interaction energy is equal to a constant $-J$ independent of the specific pair of vertices. Thus, the Hamiltonian of a state σ , or its total energy, equals $H(\sigma) = -J|U(\sigma)|$, where $U(\sigma)$ is the subset of edges with both ends of the same color. The *partition function* of the model is defined as $Z = \sum_{\sigma} e^{-H(\sigma)/kT}$, where k is the Boltzmann constant and T is the temperature. Under random interaction, the probability of finding the system in a state σ equals $\exp(-H(\sigma)/kT)/Z$.

Letting $y = e^{-J/kT}$, we can rewrite Z as a rational function of y as follows:

$$Z(y) := \sum_{\sigma} y^{-|U(\sigma)|}.$$

We intend to relate $Z(y)$ to the cocycle code of Γ . Let q be a prime power and consider the representation of the cycle matroid $M(\Gamma)$ over the field \mathbb{F}_q by the columns of a matrix \mathbf{G} . The *cocycle code* $C(\Gamma)$ [9] is the row space of \mathbf{G} . The length of $C(\Gamma)$ equals n ; the dimension is $|V| - c(E)$.

The main result of this section is given in the following theorem.

THEOREM 4.1. *Let $\mathcal{A}(x, y)$ be the weight polynomial of $C(\Gamma)$. Then*

$$\mathcal{A}(1, y) = q^{-c(E)} y^n Z(y).$$

Proof. We have the following chain of equalities:

$$\begin{aligned}
 Z(y) &= \sum_{\sigma} y^{-|U(\sigma)|} = \sum_{\sigma} \left(1 + \frac{1-y}{y}\right)^{|U(\sigma)|} = \sum_{\sigma} \sum_{F \subseteq U(\sigma)} (1-y)^{|F|} y^{-|F|} \\
 &\stackrel{(a)}{=} \sum_{F \subseteq E} (1-y)^{|F|} y^{-|F|} q^{c(F)} = \sum_{i=0}^n (1-y)^i y^{-i} \sum_{|F|=i} q^{c(F)} \\
 &\stackrel{(b)}{=} \sum_{i=0}^n (1-y)^i y^{-i} \sum_{|F|=i} q^{|V| - \text{rk}(\mathbf{G}(F))} \\
 &\stackrel{(c)}{=} q^{c(E)} \sum_{i=0}^n (1-y)^i y^{-i} \sum_{|F|=i} |C^F| \\
 &\stackrel{(d)}{=} q^{c(E)} \sum_{j=0}^n (1-y)^{n-j} y^{-(n-j)} \mathcal{B}_j \\
 &\stackrel{(e)}{=} q^{c(E)} y^{-n} \mathcal{A}(1, y),
 \end{aligned}$$

where $c(F)$ is the number of connected components in the subgraph (V, F) formed on the vertices of Γ by the edges in F . Here (a) follows by counting in two ways the size of the set

$$\{(F, \sigma) \mid F \subseteq E, \text{ connected components of } (V, F) \text{ are monochromatic}\};$$

in (b) we use the fact that the cocycle rank of the graph (V, F) equals $\text{rk}(\mathbf{G}(F)) = |V| - c(F)$; (c) follows by Lemma 2.1(i); (d) relies on (2.3); and (e) is implied by the first equality in (2.5). \square

Together with Theorem 2.2 this theorem implies the relation between the Tutte polynomial and the function Z , which is, of course, a well-known fact [11], [24, p. 64]. Therefore, although Theorem 4.1 was not explicitly stated in the literature, it can be deduced from Greene’s theorem.

Theorem 2.3 enables us relate $Z(y)$ to the rank polynomial of $C(M)$ as follows:

$$Z(y) = q^{|V|} \mathcal{U}\left(\frac{1-y}{y}, \frac{1}{q}\right).$$

This implies an interpretation of the coefficients of $Z(y)$ in terms of the number of subsets of E of a given size and rank, and, in particular, of the number f_i of independent subsets of size i .

Further connections between spin models and combinatorial theory of codes (theory of association schemes) are covered in the survey [18].

5. Concluding remark. The results of this paper can be extended from linear matroids to a somewhat broader class of almost affine matroids introduced in [22]. To define almost affine representability of a matroid M with the rank function ρ on the ground set E of size n , consider an $N \times n$ matrix D with entries from a finite set of size q . As above, for $F \subseteq E$ let $D(F)$ be the submatrix of D formed by columns with numbers in F . Let

$$r(F) = \log_q |\{\text{number of different rows in } D(F)\}|.$$

We say that M is almost affinely represented by D if $r(F)$ is integer for every $F \subseteq E$ and $\rho F = r(F)$ for every $F \subseteq E$. A matroid is called *almost affine* if it allows an almost affine representation. Linear matroids form a subset of the class of almost affine matroids; as proved in [22], this inclusion is proper.

REFERENCES

- [1] A. ASHIKHMIN, A. BARG, AND S. LITSYN, *Estimates of the distance distribution of codes and designs*, IEEE Trans. Inform. Theory, 47 (2001), pp. 1050–1061.
- [2] A. ASHIKHMIN, A. BARG, AND S. LITSYN, *Estimates of the distance distribution of nonbinary codes, with applications*, in Codes and Association Schemes, A. Barg and S. Litsyn, eds., AMS, Providence, RI, 2001, pp. 287–303.
- [3] A. BARG, *The matroid of supports of a linear code*, Appl. Algebra Engrg. Commun. Comput., 8 (1997), pp. 165–172.
- [4] A. BARG AND A. ASHIKHMIN, *Binomial moments of the distance distribution and the probability of undetected error*, Des. Codes Cryptogr., 16 (1999), pp. 103–116.
- [5] N. L. BIGGS, *Interaction Models*, London Math. Soc. Lecture Note Ser. 30, Cambridge University Press, Cambridge, UK, 1977.
- [6] T. H. BRYLAWSKI AND J. G. OXLEY, *The Tutte polynomial and its applications*, in Matroid Applications, Encyclopedia Math. Appl. 40, N. White, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 123–225.
- [7] C. J. COLBOURN AND D. D. HARMS, *Bounding all-terminal reliability in computer networks*, Networks, 18 (1988), pp. 1–12.
- [8] C. GREENE, *Weight enumeration and the geometry of linear codes*, Stud. Appl. Math., 55 (1976), pp. 119–128.
- [9] S. L. HAKIMI AND J. G. BREDESON, *Graph theoretic error-correcting codes*, IEEE Trans. Inform. Theory, 14 (1968), pp. 584–591.
- [10] T. HELLESETH, T. KLØVE, AND V. I. LEVENSHTAIN, *On the information function of an error-correcting code*, IEEE Trans. Inform. Theory, 43 (1997), pp. 549–557.
- [11] F. JAEGER, *Tutte polynomial and link polynomials*, Proc. Amer. Math. Soc., 103 (1988), pp. 647–654.
- [12] T. KLØVE AND V. I. KORZHIK, *Error Detecting Codes*, Kluwer Academic Publishers, Boston, 1995.
- [13] V. I. KORZHIK, *Bounds on the probability of undetected error and optimum group codes in a channel with feedback*, Radiotekhnika, 20 (1965), pp. 27–33 (in Russian). English translation in Telecommun. Radio Eng., 20 (1, pt.2) (1965), pp. 87–92.
- [14] V. I. LEVENSHTAIN, *Bounds on the probability of undetected error*, Problemy Peredachi Informatsii, 13 (1977), pp. 3–18.
- [15] V. I. LEVENSHTAIN, *Krawtchouk polynomials and universal bounds for codes and designs in Hamming spaces*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1303–1321.
- [16] F. J. MACWILLIAMS, *A theorem in the distribution of weights in a systematic code*, Bell System Tech. J., 42 (1963), pp. 79–94.
- [17] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, 3rd ed., North-Holland, Amsterdam, 1991.
- [18] K. NOMURA, *Spin models and Bose-Mesner algebra*, European J. Combin., 20 (1999), pp. 691–700.
- [19] V. PLESS AND W. C. HUFFMAN, EDS., *Handbook of Coding Theory*, Vol. 1, 2, Elsevier Science, Amsterdam, 1998.
- [20] V. REINER, *An interpretation for the Tutte polynomial*, European J. Combin., 20 (1999), pp. 149–161.
- [21] J. SIMONIS, *The effective length of subcodes*, Appl. Algebra Engrg. Commun. Comput., 5 (1994), pp. 371–377.
- [22] J. SIMONIS AND A. ASHIKHMIN, *Almost affine codes*, Des. Codes Cryptogr., 14 (1998), pp. 179–197.
- [23] D. J. A. WELSH, *Matroid Theory*, Academic Press, London, 1976.
- [24] D. J. A. WELSH, *Complexity: Knots, Colourings and Counting*, London Math. Soc. Lecture Notes Ser. 186, Cambridge University Press, Cambridge, UK, 1993.

SINGLE MACHINE SCHEDULING WITH RELEASE DATES*

MICHEL X. GOEMANS[†], MAURICE QUEYRANNE[‡], ANDREAS S. SCHULZ[§],
MARTIN SKUTELLA[¶], AND YAOGUANG WANG^{||}

Abstract. We consider the scheduling problem of minimizing the average weighted completion time of n jobs with release dates on a single machine. We first study two linear programming relaxations of the problem, one based on a time-indexed formulation, the other on a completion-time formulation. We show their equivalence by proving that a $O(n \log n)$ greedy algorithm leads to optimal solutions to both relaxations. The proof relies on the notion of mean busy times of jobs, a concept which enhances our understanding of these LP relaxations. Based on the greedy solution, we describe two simple randomized approximation algorithms, which are guaranteed to deliver feasible schedules with expected objective function value within factors of 1.7451 and 1.6853, respectively, of the optimum. They are based on the concept of common and independent α -points, respectively. The analysis implies in particular that the worst-case relative error of the LP relaxations is at most 1.6853, and we provide instances showing that it is at least $e/(e-1) \approx 1.5819$. Both algorithms may be derandomized; their deterministic versions run in $O(n^2)$ time. The randomized algorithms also apply to the on-line setting, in which jobs arrive dynamically over time and one must decide which job to process without knowledge of jobs that will be released afterwards.

Key words. approximation algorithm, LP relaxation, scheduling, on-line algorithm

AMS subject classifications. 90C27, 68Q25, 90B35, 68M20

PII. S089548019936223X

1. Introduction. We study the single-machine scheduling problem with release dates in which the objective is to minimize a weighted sum of completion times. It is defined as follows. A set $N = \{1, 2, \dots, n\}$ of n jobs has to be scheduled on a single disjunctive machine. Job j has a processing time $p_j > 0$ and is released at time $r_j \geq 0$. We assume that release dates and processing times are integral. The completion time of job j in a schedule is denoted by C_j . The goal is to find a nonpreemptive schedule that minimizes $\sum_{j \in N} w_j C_j$, where the w_j 's are given positive weights. In the classical scheduling notation [12], this problem is denoted by $1|r_j|\sum w_j C_j$. It is strongly NP-hard, even if $w_j = 1$ for all jobs j [17].

*Received by the editors October 1, 1999; accepted for publication (in revised form) November 15, 2001; published electronically February 27, 2002.

<http://www.siam.org/journals/sidma/15-2/36223.html>

[†]Department of Mathematics, Room 2-351, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139 (goemans@math.mit.edu). The research of this author was performed partly when the author was at C.O.R.E., Louvain-la-Neuve, Belgium and was supported in part by NSF contract 9623859-CCR.

[‡]Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, BC, Canada V6T 1Z2 (maurice.queyranne@commerce.ubc.ca). The research of this author was supported in part by a research grant from the NSERC (Natural Sciences and Research Council of Canada) and by the UNI.TU.RIM. S.p.a. (Società per l'Università nel riminese).

[§]Sloan School of Management, Room E53-361, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139 (schulz@mit.edu). The research of this author was performed partly when he was with the Department of Mathematics, Technische Universität Berlin, Germany.

[¶]Fakultät II—Mathematik und Naturwissenschaften, Institut für Mathematik, MA 6-1, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany (skutella@math.tu-berlin.de). This author was supported in part by DONET within the frame of the TMR Programme (contract ERB FMRX-CT98-0202) while staying at C.O.R.E., Louvain-la-Neuve, Belgium, for the academic year 1998/99.

^{||}Peoplesoft, Inc., Pleasanton, CA 94588 (Yaoguang.Wang@peoplesoft.com). This author was supported by a research fellowship from Max-Planck Institute for Computer Science, Saarbrücken, Germany.

One of the key ingredients in the design and analysis of approximation algorithms as well as in the design of implicit enumeration methods is the choice of a bound on the optimal value. Several linear programming-based as well as combinatorial lower bounds have been proposed for this well-studied scheduling problem; see, for example, Dyer and Wolsey [9], Queyranne [22], and Queyranne and Schulz [23], as well as Belouadah, Posner, and Potts [4]. The LP relaxations involve a variety of different types of variables which, e.g., express whether either job j is completed at time t (nonpreemptive time-indexed relaxation), or whether it is being processed at time t (preemptive time-indexed relaxation), or when job j is completed (completion time relaxation). Dyer and Wolsey show that the nonpreemptive time-indexed relaxation is stronger than the preemptive time-indexed relaxation. We will show that the latter relaxation is equivalent to the completion time relaxation that makes use of the so-called shifted parallel inequalities. In fact, it turns out that the polyhedron defined by these inequalities is supermodular, and hence one can optimize over it by using the greedy algorithm. A very similar situation arises in [24]. The greedy solution may actually be interpreted in terms of the following preemptive schedule, which we call the LP schedule: at any point in time it schedules among the available jobs one with the largest ratio of weight to processing time. Uma and Wein [38] point out that the value of this LP solution coincides with one of the combinatorial bounds of Belouadah, Posner, and Potts based on the idea of allowing jobs to be split into smaller pieces that can be scheduled individually.

We show that the optimal value of $1|r_j|\sum w_jC_j$ is at most 1.6853 times the lower bound given by any of these three equivalent relaxations—the preemptive time-indexed relaxation, the completion time relaxation, or the combinatorial relaxation in [4]. We prove this result on the quality of these relaxations by converting the (preemptive) LP schedule into a nonpreemptive schedule. This technique leads to approximation algorithms for $1|r_j|\sum w_jC_j$. Recall that a ρ -approximation algorithm is a polynomial-time algorithm guaranteed to deliver a solution of cost at most ρ times the optimal value. A *randomized* ρ -approximation algorithm is a polynomial-time algorithm that produces a feasible solution whose *expected* objective function value is within a factor of ρ of the optimal value.

The technique of converting preemptive schedules to nonpreemptive schedules in the design of approximation algorithms was introduced by Phillips, Stein, and Wein [21]. More specifically, for $1|r_j|\sum w_jC_j$ they showed that list scheduling in order of the completion times of a given preemptive schedule produces a nonpreemptive schedule while increasing the total weighted completion time by at most a factor of 2. In the same paper they also introduced a concept of α -points. This notion was also used by Hall, Shmoys, and Wein [14] in connection with the nonpreemptive time-indexed relaxation of Dyer and Wolsey to design approximation algorithms in various scheduling environments. For our purposes, the α -point of job j in a given preemptive schedule is the first point in time at which an α -fraction of j has been completed. When one chooses different values of α , sequencing in order of nondecreasing α -points in the same preemptive schedule may lead to different nonpreemptive schedules. This increased flexibility led to improved approximation algorithms: Chekuri et al. [6] for $1|r_j|\sum C_j$ and Goemans [11] for $1|r_j|\sum w_jC_j$ chose α at random and analyzed the expected performance of the resulting randomized algorithms. We will show that, using a common value of α for all jobs and an appropriate probability distribution, sequencing in order of α -points of the LP schedule has expected performance no worse than 1.7451 times the optimal preemptive time-indexed LP value. We also prove that

TABLE 1

Summary of approximation bounds for $1|r_j|\sum w_j C_j$. An α -schedule is obtained by sequencing the jobs in order of nondecreasing α -points of the LP schedule. The use of job-dependent α_j 's yields an (α_j) -schedule. The results discussed in this paper are below the second double line. Subsequently, Anderson and Potts [2] gave a deterministic 2-competitive algorithm. For the unit-weight problem $1|r_j|\sum C_j$, the first constant-factor approximation algorithm is due to Phillips, Stein, and Wein [21]. It has performance ratio 2, and it also works on-line. Further deterministic 2-competitive algorithms were given by Stogie [36] and Hoogeveen and Vestjens [15]. All these algorithms are optimal for deterministic on-line algorithms [15]. Chekuri et al. [6] gave a randomized $e/(e-1)$ -approximation algorithm, which is optimal for randomized on-line algorithms [37, 39].

Reference and/or type of schedule	Off-line deterministic	On-line	
		randomized	deterministic
Phillips et al. [21]	$16 + \epsilon$		
Hall et al. [14]	4		$4 + \epsilon$
Schulz [26]	3		
Hall et al. [13]	3		$3 + \epsilon$
Chakrabarti et al. [5]	$2.8854 + \epsilon$	$2.8854 + \epsilon$	
Combining [5] and [13]	$2.4427 + \epsilon$	$2.4427 + \epsilon$	
α -schedule for $\alpha = 1/\sqrt{2}$ [11]	2.4143		2.4143
BEST α -schedule [11]	2		
	1.7451		
(random) (α_j) -schedule	1.6853	1.6853	

by selecting a separate value α_j for each job j , one can improve this bound to a factor of 1.6853. Our algorithms are inspired by and partly resemble the algorithms of Hall, Shmoys, and Wein [14] and Chekuri et al. [6]. In contrast to Hall, Shmoys, and Wein we exploit the preemptive time-indexed LP relaxation, which, on the one hand, provides us with highly structured optimal solutions and, on the other hand, enables us to work with mean busy times. We also use random α -points. The algorithm of Chekuri et al. starts from an arbitrary preemptive schedule and makes use of random α -points. They relate the value of the resulting α -schedule to that of the given preemptive schedule and not to that of an underlying LP relaxation. While their approach gives better approximations for $1|r_j|\sum C_j$ and structural insights on limits of the power of preemption, the link of the LP schedule to the preemptive time-indexed LP relaxation helps us to obtain good approximations for the total weighted completion time.

Variants of our algorithms also work on-line when jobs arrive dynamically over time and, at each point in time, one has to decide which job to process without knowledge of jobs that will be released afterwards. Even in this on-line setting, we compare the value of the computed schedule to the optimal (off-line) schedule and derive the same bounds (competitive ratios) as in the off-line setting. See Table 1 for an account of the evolution of off-line and on-line approximation results for the single machine problem under consideration.

The main ingredient to obtain the results presented in this paper is the exploitation of the structure of the LP schedule. Not surprisingly, the LP schedule does not solve the strongly NP-hard [16] preemptive version of the problem, $1|r_j, pmtn|\sum w_j C_j$. However, we show that the LP schedule solves optimally the preemptive problem with the related objective function $\sum_j w_j M_j$, where M_j is the mean busy time of job j , i.e., the average point in time at which the machine is busy processing j . Observe that, although $1|r_j, pmtn|\sum w_j C_j$ and $1|r_j, pmtn|\sum w_j M_j$ are equivalent optimization problems in the nonpreemptive case (since $C_j = M_j + \frac{p_j}{2}$),

they are not equivalent when considering preemptive schedules.

The approximation techniques presented in this paper have also proved useful for more general scheduling problems. For the problem with precedence constraints $1|r_j, prec|\sum w_j C_j$, sequencing jobs in order of random α -points based on an optimal solution to a time-indexed LP relaxation leads to a 2.7183-approximation algorithm [27]. A 2-approximation algorithm for identical parallel machine scheduling $P|r_j|\sum w_j C_j$ is given in [28]; the result is based on a time-indexed LP relaxation, an optimal solution of which can be interpreted as a preemptive schedule on a fast single machine; jobs are then assigned randomly to the machines and sequenced in order of random α_j -points of this preemptive schedule. For the corresponding scheduling problem on unrelated parallel machines $R|r_j|\sum w_j C_j$, a performance guarantee of 2 can be obtained by randomized rounding based on a convex quadratic programming relaxation [33], which is inspired by time-indexed LP relaxations like the one discussed herein [28]. We refer to [32] for a detailed discussion of the use of α -points for machine scheduling problems.

Significant progress has recently been made in understanding the approximability of scheduling problems with the average weighted completion time objective. Skutella and Woeginger [34] developed a polynomial-time approximation scheme for scheduling identical parallel machines in the absence of release dates, $P||\sum w_j C_j$. Subsequently, several research groups have found polynomial-time approximation schemes for problems with release dates such as $P|r_j|\sum w_j C_j$ and $Rm|r_j|\sum w_j C_j$; see the resulting joint conference proceedings publication [1] for details.

We now briefly discuss some practical consequences of our work. Savelsbergh, Uma, and Wein [25] and Uma and Wein [38] performed experimental studies to evaluate, in part, the quality of the LP relaxation and approximation algorithms studied herein for $1|r_j|\sum w_j C_j$ and related scheduling problems. The first authors report that, except for instances that were deliberately constructed to be hard for this approach, the present formulation and algorithms “deliver surprisingly strong experimental performance.” They also note that “the ideas that led to improved approximation algorithms also lead to heuristics that are quite effective in empirical experiments; furthermore [...] they can be extended to give improved heuristics for more complex problems that arise in practice.” While the authors of the follow-up study [38] report that when coupled with local improvement the LP-based heuristics generally produce the best solutions, they also find that a simple heuristic often outperforms the LP-based heuristics. Whenever the machine becomes idle, this heuristic starts nonpreemptively processing an available job of largest w_j/p_j ratio. By analyzing the differences between the LP schedule and this heuristic schedule, Chou, Queyranne, and Simchi-Levi [7] have subsequently shown the asymptotic optimality of this on-line heuristic for classes of instances with bounded job weights and bounded processing times.

The contents of this paper are as follows. Section 2 is concerned with the LP relaxations and their relationship. We begin with a presentation and discussion of the LP schedule. In section 2.1 we then review a time-indexed formulation introduced by Dyer and Wolsey [9] and show that it is solved to optimality by the LP schedule. In section 2.2 we present the mean busy time relaxation (or completion time relaxation) and prove, among other properties, its equivalence to the time-indexed formulation. Section 2.3 explores some polyhedral consequences, in particular the fact that the mean busy time relaxation is (up to scaling by the job processing times) a super-modular linear program and that the “job-based” method for constructing the LP

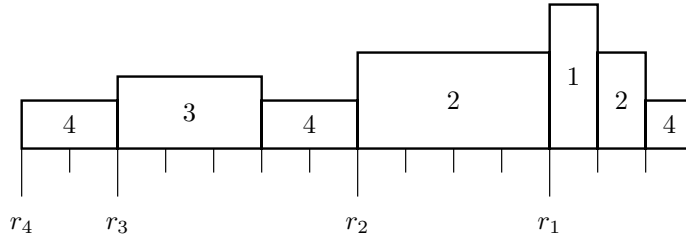


FIG. 1. An LP schedule for a 4-job instance given by $r_1 = 11$, $p_1 = 1$, $r_2 = 7$, $p_2 = 5$, $r_3 = 2$, $p_3 = 3$, $r_4 = 0$, $p_4 = 5$. Higher rectangles represent jobs with larger weight to processing time ratio. Time is shown on the horizontal axis.

schedule is equivalent to the corresponding greedy algorithm. Section 3 then deals with approximation algorithms derived from these LP relaxations. In section 3.1 we present a method for constructing (α_j) -schedules, which allows us to analyze and bound the job completion times in the resulting schedules. In section 3.2 we derive simple bounds for α -schedules and (α_j) -schedules, using a deterministic common α or uniformly distributed random α_j 's. Using appropriate probability distributions, we improve the approximation bound to the value of 1.7451 for α -schedules in section 3.3 and to the value of 1.6853 for (α_j) -schedules in section 3.4. We also indicate how these algorithms can be derandomized in $O(n^2)$ time for constructing deterministic schedules with these performance guarantees. In section 3.5 we show that our randomized approximations also apply in an on-line setting, and in section 3.6 we present a class of “bad” instances for which the ratio of the optimal objective function value and our LP bound is arbitrarily close to $\frac{e}{e-1} \approx 1.5819$. This constant defines a lower bound on the approximation results that can be obtained by the present approach. We conclude in section 4 by discussing some related problems and open questions.

2. Relaxations. In this section, we present two linear programming relaxations for $1|r_j|\sum w_j C_j$. We show their equivalence and discuss some polyhedral consequences.

For both relaxations, the following preemptive schedule plays a crucial role: at any point in time, schedule (preemptively) the available job with the highest w_j/p_j ratio. We assume (throughout the paper) that the jobs are indexed in order of nonincreasing ratios $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \dots \geq \frac{w_n}{p_n}$, and ties are broken according to this order. Therefore, whenever a job is released, the job being processed (if any) is preempted if the released job has a smaller index. We refer to this preemptive schedule as *the LP schedule*. See Figure 1 for an example of an LP schedule.

Notice that this LP schedule does not in general minimize $\sum_j w_j C_j$ over all preemptive schedules. This should not be surprising since the preemptive problem $1|r_j, pmtn|\sum w_j C_j$ is (strongly) NP-hard [16]. It can be shown, however, that the total weighted completion time of the LP schedule is always within a factor of 2 of the optimal value for $1|r_j, pmtn|\sum w_j C_j$, and this bound is tight; see [29].

The LP schedule can be constructed in $O(n \log n)$ time. To see this, we now describe an implementation, which may be seen as “dynamic” (event-oriented) or, using the terminology of [19], “machine-based,” and can even be executed on-line while the jobs dynamically arrive over time. The algorithm keeps a priority queue [8] of the currently available jobs that have not yet been completely processed, with the ratio w_j/p_j as the key and with another field indicating the remaining processing time. A scheduling decision is made at only two types of events: when a job is released and

when a job completes its processing. In the former case, the released job is added to the priority queue. In the latter case, the completed job is removed from the priority queue. Then, in either case, the top element of the priority queue (the one with the highest w_j/p_j ratio) is processed; if the queue is empty, then move on to the next job release; if there is none, then all jobs have been processed and the LP schedule is complete. This implementation results in a total of $O(n)$ priority queue operations. Since each such operation can be implemented in $O(\log n)$ time [8], the algorithm runs in $O(n \log n)$ time.

The LP schedule can also be defined in a somewhat different manner, which may be seen as “static” or “job-based” [19]. Consider the jobs one at a time in order of nonincreasing w_j/p_j . Schedule each job j as early as possible starting at r_j and preempting it whenever the machine is busy processing another job (that thus came earlier in the w_j/p_j ordering). This point of view leads to an alternate $O(n \log n)$ construction of the LP schedule; see [10].

2.1. Time-indexed relaxation. Dyer and Wolsey [9] investigate several types of relaxations of $1|r_j|\sum w_j C_j$, the strongest ones being time-indexed. We consider the weaker of their two time-indexed formulations, which they call formulation (D). It uses two types of variables: $y_{j\tau} = 1$ if job j is being processed during time interval $[\tau, \tau + 1)$, and zero otherwise; and t_j represents the start time of job j . For simplicity, we add p_j to t_j and replace the resulting expression by C_j ; this gives an equivalent relaxation.

$$\begin{aligned}
 Z_D &= \min \sum_{j \in N} w_j C_j \\
 &\text{subject to} \\
 (D) \quad &\sum_{j: r_j \leq \tau} y_{j\tau} \leq 1, & \tau = 0, 1, \dots, T-1, \\
 &\sum_{\tau=r_j}^{T-1} y_{j\tau} = p_j, & j \in N, \\
 (2.1) \quad &C_j = \frac{1}{2}p_j + \frac{1}{p_j} \sum_{\tau=r_j}^{T-1} \left(\tau + \frac{1}{2}\right) y_{j\tau}, & j \in N, \\
 &0 \leq y_{j\tau}, & j \in N, \tau = r_j, \dots, T-1,
 \end{aligned}$$

where T is an upper bound on the makespan of an optimal schedule. (For example, $T = \max_{j \in N} r_j + \sum_{j \in N} p_j$.) We refer to this relaxation as the *preemptive time-indexed relaxation*. The expression for C_j given in (2.1) corresponds to the correct value of the completion time if job j is not preempted; an interpretation in terms of mean busy times is given in the next section for the case of preemptions. Observe that the number of variables of this formulation is pseudopolynomial. If we eliminate C_j from the relaxation by using (2.1), we obtain a transportation problem [9] and, as a result, $y_{j\tau}$ can be assumed to be integral.

LEMMA 2.1. *There exists an optimal solution to (D) for which $y_{j\tau} \in \{0, 1\}$ for all j and τ .*

As indicated in [9], (D) can be solved in $O(n \log n)$ time. Actually, one can derive a feasible solution to (D) from the LP schedule by letting $y_{j\tau}^{LP}$ be equal to 1 if job j is being processed in $[\tau, \tau + 1)$, and 0 otherwise.

THEOREM 2.2. *The solution y^{LP} derived from the LP schedule is an optimal solution to (D).*

Proof. The proof is based on an interchange argument. Consider any optimal 0/1-solution y^* to (D). If there exist $j < k$ and $\sigma > \tau \geq r_j$ such that $y_{j\sigma}^* = y_{k\tau}^* = 1$, then by replacing $y_{j\sigma}^*$ and $y_{k\tau}^*$ by 0, and $y_{j\tau}^*$ and $y_{k\sigma}^*$ by 1, we obtain another feasible solution with an increase in the objective function value of $(\sigma - \tau)(\frac{w_k}{p_k} - \frac{w_j}{p_j}) \leq 0$. The resulting solution must therefore also be optimal. By repeating this interchange argument, we derive that there exists an optimal solution y^* such that there do not exist $j < k$ and $\sigma > \tau \geq r_j$ such that $y_{j\sigma}^* = y_{k\tau}^* = 1$. This implies that the solution y^* must correspond to the LP schedule. \square

In particular, despite the pseudopolynomial number of variables in the LP Relaxation (D) an optimal solution can be obtained efficiently. We will make use of this fact as well as of the special structure of the LP schedule in the design and analysis of the approximation algorithms; see section 3. We note again that in spite of its nice properties the preemptive time-indexed LP Relaxation (D) solves neither $1|r_j|\sum w_j C_j$ nor $1|r_j, pmtn|\sum w_j C_j$. In the former case, the processing of a job in the LP solution may fail to be consecutive; in the latter case (2.1) does not necessarily define the completion time of a job in the preemptive LP schedule, as will be shown in the next lemma.

2.2. Mean busy time relaxation. Given any preemptive schedule, let I_j be the indicator function of the processing of job j at time t ; i.e., $I_j(t)$ is 1 if the machine is processing j at time t , and 0 otherwise. To avoid pathological situations, we require that, in any preemptive schedule, when the machine starts processing a job, it does so for a positive amount of time. Given any preemptive schedule, we define the *mean busy time* M_j of job j to be the average time at which the machine is processing j , that is,

$$M_j := \frac{1}{p_j} \int_{r_j}^T I_j(t) t dt.$$

For instance, in the example given in Figure 1, which will be used throughout this paper, the mean busy time of job 4 is 5.5.

We first establish some important properties of M_j in the next two lemmas.

LEMMA 2.3. *For any preemptive schedule, let C_j and M_j denote the completion and mean busy time, respectively, of job j . Then for any job j , we have $M_j + \frac{1}{2}p_j \leq C_j$, with equality if and only if job j is not preempted.*

Proof. If job j is processed without preemption, then $I_j(t) = 1$ if $C_j - p_j \leq t < C_j$, and 0 otherwise; therefore, $M_j + \frac{1}{2}p_j = C_j$. Otherwise, job j is not processed during some interval(s) of total length $L > 0$ between times $C_j - p_j$ and C_j . Since $\int_{r_j}^T I_j(t) dt = p_j$, job j must be processed during some time interval(s) of the same total length L before $C_j - p_j$. Therefore,

$$M_j = \frac{1}{p_j} \int_{r_j}^{C_j} I_j(t) t dt < \frac{1}{p_j} \int_{C_j-p_j}^{C_j} t dt = C_j - \frac{1}{2}p_j$$

and the proof is complete. \square

Let $S \subseteq N$ denote a set of jobs and define

$$p(S) := \sum_{j \in S} p_j \quad \text{and} \quad r_{\min}(S) := \min_{j \in S} r_j.$$

Let $I_S(t) := \sum_{j \in S} I_j(t)$. Since, by the machine capacity constraint, $I_S(t) \in \{0, 1\}$ for all t , we may view I_S as the indicator function for job set S . We can thus define the *mean busy time of set S* as $M_S := \frac{1}{p(S)} \int_0^T I_S(t) t dt$. Note that we have

$$(2.2) \quad p(S)M_S = \int_0^T \left(\sum_{j \in S} I_j(t) \right) t dt = \sum_{j \in S} \int_0^T I_j(t) t dt = \sum_{j \in S} p_j M_j.$$

So, unlike its start and completion time, the mean busy time of a job set is a simple weighted average of the mean busy times of its elements. One consequence of this observation is the validity of the *shifted parallel inequalities* (2.3) (see, e.g., [10, 23, 24]) below.

LEMMA 2.4. *For any set S of jobs and any preemptive schedule with mean busy time vector M , we have*

$$(2.3) \quad \sum_{j \in S} p_j M_j \geq p(S) \left(r_{\min}(S) + \frac{1}{2}p(S) \right),$$

and equality holds if and only if all jobs in S are scheduled without interruption from $r_{\min}(S)$ to $r_{\min}(S) + p(S)$.

Proof. Note that $\sum_{j \in S} p_j M_j = p(S)M_S = \int_{r_{\min}(S)}^T I_S(t) t dt$, that $I_S(t) = 0$ for $t < r_{\min}(S)$ and $I_S(t) \leq 1$ for $t \geq r_{\min}(S)$, and that $\int_{r_{\min}(S)}^T I_S(t) dt = p(S)$. Under these constraints, M_S is minimized when $I_S(t) = 1$ for $r_{\min}(S) \leq t < r_{\min}(S) + p(S)$, and 0 otherwise. Therefore, M_S is uniquely minimized among all feasible preemptive schedules when all jobs in S are continuously processed from $r_{\min}(S)$ to $r_{\min}(S) + p(S)$. This minimal value is $p(S)(r_{\min}(S) + \frac{1}{2}p(S))$ and is a lower bound for $\sum_{j \in S} p_j M_j$ in any feasible preemptive schedule. \square

As a result of Lemma 2.4, the following linear program provides a lower bound on the optimal value of $1|r_j, pmtn | \sum w_j C_j$, and hence on that of $1|r_j | \sum w_j C_j$.

$$(R) \quad \begin{aligned} Z_R &= \min \sum_{j \in N} w_j \left(M_j + \frac{1}{2}p_j \right) \\ &\text{subject to} \\ &\sum_{j \in S} p_j M_j \geq p(S) \left(r_{\min}(S) + \frac{1}{2}p(S) \right), \quad S \subseteq N. \end{aligned}$$

The proof of the following theorem and later developments use the notion of canonical decompositions [10]. For a set S of jobs, consider the schedule which processes jobs in S as early as possible, say, in order of their release dates. This schedule induces a partition of S into $\{S_1, \dots, S_k\}$ such that the machine is busy processing jobs in S exactly in the disjoint intervals $[r_{\min}(S_\ell), r_{\min}(S_\ell) + p(S_\ell)]$ for $\ell = 1, \dots, k$. We refer to this partition as the *canonical decomposition* of S . A set is *canonical* if it is identical to its canonical decomposition, i.e., if its canonical decomposition is $\{S\}$. Thus a set S is canonical if and only if it is feasible to schedule all its jobs in the time interval $[r_{\min}(S), r_{\min}(S) + p(S))$. Note that the set $N = \{1, 2, 3, 4\}$ in our example is canonical, whereas the subset $\{1, 2, 3\}$ is not; it has the decomposition $\{\{3\}, \{1, 2\}\}$. Let

$$(2.4) \quad h(S) := \sum_{\ell=1}^k p(S_\ell) \left(r_{\min}(S_\ell) + \frac{1}{2}p(S_\ell) \right),$$

where $\{S_1, \dots, S_k\}$ is the canonical decomposition of $S \subseteq N$. Then Lemma 2.4 implies that $\sum_{j \in S} p_j M_j = \sum_{\ell=1}^k \sum_{j \in S_\ell} p_j M_j \geq h(S)$ is a valid inequality for the mean busy time vector of any preemptive schedule. In other words, Relaxation (R) may be written as

$$\min \left\{ \sum_{j \in N} w_j \left(M_j + \frac{1}{2} p_j \right) : \sum_{j \in S} p_j M_j \geq h(S) \text{ for all } S \subseteq N \right\}.$$

THEOREM 2.5. *Let M_j^{LP} be the mean busy time of job j in the LP schedule. Then M^{LP} is an optimal solution to (R).*

Proof. By Lemma 2.4, M^{LP} is a feasible solution for (R).

To prove optimality of M^{LP} , we construct a lower bound on the optimal value of (R) and show that it is equal to the objective function value of M^{LP} . Recall that the jobs are indexed in nonincreasing order of the w_j/p_j ratios; let $[i] := \{1, 2, \dots, i\}$, and let $S_1^i, \dots, S_{k(i)}^i$ denote the canonical decomposition of $[i]$. Observe that for any vector $M = (M_j)_{j \in N}$ we have

$$(2.5) \quad \sum_{j \in N} w_j M_j = \sum_{i=1}^n \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{j \in [i]} p_j M_j = \sum_{i=1}^n \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{\ell=1}^{k(i)} \sum_{j \in S_\ell^i} p_j M_j,$$

where we let $w_{n+1}/p_{n+1} := 0$. We have therefore expressed $\sum_{j \in N} w_j M_j$ as a non-negative combination of expressions $\sum_{j \in S_\ell^i} p_j M_j$ over canonical sets. By construction of the LP schedule, the jobs in any of these canonical sets S_ℓ^i are continuously processed from $r_{\min}(S_\ell^i)$ to $r_{\min}(S_\ell^i) + p(S_\ell^i)$ in the LP schedule. Thus, for any feasible solution M to (R) and any such canonical set S_ℓ^i we have

$$\sum_{j \in S_\ell^i} p_j M_j \geq h(S_\ell^i) = p(S_\ell^i) \left(r_{\min}(S_\ell^i) + \frac{1}{2} p(S_\ell^i) \right) = \sum_{j \in S_\ell^i} p_j M_j^{LP},$$

where the last equation follows from Lemma 2.4. Combining this with (2.5), we derive a lower bound on $\sum_j w_j M_j$ for any feasible solution M to (R), and this lower bound is attained by the LP schedule. \square

From Theorems 2.2 and 2.5, we derive that the values of the two Relaxations (D) and (R) are equal.

COROLLARY 2.6. *The LP Relaxations (D) and (R) of $1|r_j| \sum w_j C_j$ yield the same optimal objective function value, i.e., $Z_D = Z_R$, for any weights $w \geq 0$. This value can be computed in $O(n \log n)$ time.*

Proof. For the equivalence of the two lower bounds, note that the mean busy time M_j^{LP} of any job j in the LP schedule can be expressed as

$$(2.6) \quad M_j^{LP} = \frac{1}{p_j} \sum_{\tau=r_j}^{T-1} y_{j\tau}^{LP} \left(\tau + \frac{1}{2} \right),$$

where y^{LP} is the solution to (D) derived from the LP schedule. The result then follows directly from Theorems 2.2 and 2.5. We have shown earlier that the LP schedule can be constructed in $O(n \log n)$ time. \square

Although the LP schedule does not necessarily minimize the objective function $\sum_j w_j C_j$ over the preemptive schedules, Theorem 2.5 implies that it minimizes

$\sum_j w_j M_j$ over the preemptive schedules. In addition, by Lemma 2.3, the LP schedule is also optimal for both preemptive and nonpreemptive problems $1|r_j, pmtn|\sum w_j C_j$, and $1|r_j|\sum w_j C_j$ whenever it does not preempt any job. For example, this is the case if all processing times are equal to 1 or if all jobs are released at the same date. Thus, the LP schedule provides an optimal solution to problems $1|r_j, p_j = 1|\sum w_j C_j$ and to $1|\sum w_j C_j$. This was already known. In the latter case it coincides with Smith’s ratio rule [35]; see Queyranne and Schulz [24] for the former case.

2.3. Polyhedral consequences. We now consider some polyhedral consequences of the preceding results. Let P_D^∞ be the feasible region defined by the constraints of Relaxation (D) when $T = \infty$, i.e.,

$$P_D^\infty := \left\{ y \geq 0 : \sum_{j:r_j \leq \tau} y_{j\tau} \leq 1 \text{ for } \tau \in \mathbb{N}; \sum_{\tau \geq r_j} y_{j\tau} = p_j \text{ for all } j \in N \right\}.$$

In addition, we denote by $P_R := \{M \in \mathbb{R}^N : \sum_{j \in S} p_j M_j \geq h(S) \text{ for all } S \subseteq N\}$ the polyhedron defined by the constraints of Relaxation (R).

THEOREM 2.7.

- (i) Polyhedron P_R is the convex hull of the mean busy time vectors M of all preemptive schedules. Moreover, every vertex of P_R is the mean busy time vector of an LP schedule.
- (ii) Polyhedron P_R is also the image of P_D^∞ in the space of the M -variables under the linear mapping $\mathcal{M} : y \mapsto \mathcal{M}(y) \in \mathbb{R}^N$ defined by

$$\mathcal{M}(y)_j = \frac{1}{p_j} \sum_{\tau \geq r_j} y_{j\tau} \left(\tau + \frac{1}{2} \right) \quad \text{for all } j \in N.$$

Proof. (i) Lemma 2.4 implies that the convex hull of the mean busy time vectors M of all feasible preemptive schedules is contained in P_R . To show the reverse inclusion, it suffices to show that (a) every extreme point of P_R corresponds to a preemptive schedule; and (b) every extreme ray of P_R is a direction of recession for the convex hull of mean busy time vectors. Property (a) and the second part of statement (i) follow from Theorem 2.5 and the fact that every extreme point of P_R is the unique minimizer of $\sum_{j \in N} w_j M_j$ for some $w \geq 0$. For (b), note that the extreme rays of P_R are the n unit vectors of \mathbb{R}^N . An immediate extension to preemptive schedules and mean busy times of results in Balas [3] implies that these unit vectors of \mathbb{R}^N are directions of recession for the convex hull of mean busy time vectors. This completes the proof of (i).

(ii) We first show that the image $\mathcal{M}(P_D^\infty)$ of P_D^∞ is contained in P_R . For this, let y be a vector in P_D^∞ and $S \subseteq N$ with canonical decomposition $\{S_1, \dots, S_k\}$. By definition of $\mathcal{M}(y)_j$, we have

$$\begin{aligned} \sum_{j \in S} p_j \mathcal{M}(y)_j &= \sum_{j \in S} \sum_{\tau \geq r_j} y_{j\tau} \left(\tau + \frac{1}{2} \right) \\ &\geq \sum_{\ell=1}^k \sum_{\tau=r_{\min}(S_\ell)}^{r_{\min}(S_\ell)+p(S_\ell)-1} \left(\tau + \frac{1}{2} \right) \\ &= \sum_{\ell=1}^k p(S_\ell) \left(r_{\min}(S_\ell) + \frac{1}{2} p(S_\ell) \right) = h(S). \end{aligned}$$

The inequality follows from the constraints defining P_D^∞ and the interchange argument which we already used in the proof of Theorem 2.2. This shows $\mathcal{M}(y) \in P_R$ and thus $\mathcal{M}(P_D^\infty) \subseteq P_R$.

To show the reverse inclusion, we use the observation from the proof of part (i) that P_R can be represented as the sum of the convex hull of the mean busy time vectors of all LP schedules and the nonnegative orthant. Since, by (2.6), the mean busy time vector M^{LP} of any LP schedule is the projection of the corresponding 0/1-vector y^{LP} , it remains to show that every unit vector e_j is a direction of recession for $\mathcal{M}(P_D^\infty)$. For this, fix an LP schedule and let y^{LP} and $M^{LP} = \mathcal{M}(y^{LP})$ denote the associated 0/1 y -vector and mean busy time vector, respectively. For any job $j \in N$ and any real $\lambda > 0$, we need to show that $M^{LP} + \lambda e_j \in \mathcal{M}(P_D^\infty)$.

Let $\tau_{\max} := \operatorname{argmax}\{y_{k\tau}^{LP} = 1 : k \in N\}$. Choose θ such that $y_{j\theta}^{LP} = 1$, choose an integer $\mu > \max\{\lambda p_j, \tau_{\max} - \theta\}$, and define y' by $y'_{j\theta} = 0$, $y'_{j,\theta+\mu} = 1$, and $y'_{k\tau} = y_{k\tau}^{LP}$ otherwise. In the associated preemptive schedule, the processing of job j that was done in interval $[\theta, \theta + 1)$ is now postponed, by μ time units, until interval $[\theta + \mu, \theta + \mu + 1)$. Therefore, its mean busy time vector $M' = \mathcal{M}(y')$ satisfies $M'_j = M_j^{LP} + \mu/p_j$ and $M'_k = M_k^{LP}$ for all $k \neq j$. Let $\lambda' := \mu/p_j \geq \lambda$, so $M' = M^{LP} + \lambda' e_j$. Then the vector $M^{LP} + \lambda e_j$ is a convex combination of $M^{LP} = \mathcal{M}(y^{LP})$ and $M' = \mathcal{M}(y')$. Let y be the corresponding convex combination of y^{LP} and y' . Since P_D^∞ is convex, then $y \in P_D^\infty$ and, since the mapping \mathcal{M} is linear, $M^{LP} + \lambda e_j = \mathcal{M}(y) \in \mathcal{M}(P_D^\infty)$. \square

In view of earlier results for single machine scheduling with identical release dates [22], as well as for parallel machine scheduling with unit processing times and integer release dates [24], it is interesting to note that the feasible set P_R of the mean busy time relaxation is, up to scaling by the job processing times, a supermodular polyhedron.

PROPOSITION 2.8. *The set function h defined in (2.4) is supermodular.*

Proof. Consider any two elements $j, k \in N$ and any subset $S \subseteq N \setminus \{j, k\}$. We may construct an LP schedule minimizing $\sum_{i \in S \cup \{k\}} p_i M_i$ using the job-based method by considering first the jobs in S and then job k . (Note that considering the jobs in any sequence leads to a schedule minimizing $\sum_i p_i M_i$ because jobs are weighted by their processing times in this objective function.) By Definition (2.4) the resulting mean busy times M^{LP} satisfy $\sum_{i \in S} p_i M_i^{LP} = h(S)$ and $\sum_{i \in S \cup \{k\}} p_i M_i^{LP} = h(S \cup \{k\})$. Note that job k is scheduled, no earlier than its release date, in the first p_k units of idle time left after the insertion of all jobs in S . Thus M_k^{LP} is the mean of all these p_k time units. Similarly, we may construct an LP schedule, whose mean busy time vector will be denoted by \widetilde{M}^{LP} , minimizing $\sum_{i \in S \cup \{j, k\}} p_i M_i$ by considering first the jobs in S , so $\widetilde{M}_i^{LP} = M_i^{LP}$ for all $i \in S$; then job j , so $\sum_{i \in S \cup \{j\}} p_i \widetilde{M}_i^{LP} = h(S \cup \{j\})$; and then job k , so $\sum_{i \in S \cup \{j, k\}} p_i \widetilde{M}_i^{LP} = h(S \cup \{j, k\})$. Since job j has been inserted after subset S was scheduled, job k cannot use any idle time interval that is earlier than those it used in the former LP schedule M^{LP} —and some of the previously available idle time may now be occupied by job j , causing a delay in the mean busy time of job k ; thus we have $\widetilde{M}_k^{LP} \geq M_k^{LP}$ and therefore

$$h(S \cup \{j, k\}) - h(S \cup \{j\}) = p_k \widetilde{M}_k^{LP} \geq p_k M_k^{LP} = h(S \cup \{k\}) - h(S).$$

This suffices to establish that h is supermodular. \square

An alternate proof of the supermodularity of h can be derived, as in [10], from the fact, observed by Dyer and Wolsey and already mentioned above, that Relaxation (D) becomes a transportation problem after elimination of the C_j 's. Indeed, from an

interpretation of Nemhauser, Wolsey, and Fisher [20] of a result of Shapley [31], it then follows that the value of this transportation problem as a function of S is supermodular. One of the consequences of Proposition 2.8 is that the job-based method to construct an LP schedule is just a manifestation of the greedy algorithm for minimizing $\sum_{j \in N} w_j M_j$ over the supermodular polyhedron P_R .

We finally note that the separation problem for the polyhedron P_R can be solved combinatorially. One can separate over the family of inequalities $\sum_{j \in S} p_j M_j \geq p(S)(r_{\min}(S) + p(S))$ by trying all possible values for $r_{\min}(S)$ (of which there are at most n) and then applying a $O(n \log n)$ separation routine of Queyranne [22] for the problem without release dates. The overall separation routine can be implemented in $O(n^2)$ time by observing that the bottleneck step in Queyranne's algorithm—sorting the mean busy times of the jobs—needs to be done only once for the whole job set.

3. Provably good schedules and LP relaxations. In this section, we derive approximation algorithms for $1|r_j|\sum w_j C_j$ that are based on converting the preemptive LP schedule into a feasible nonpreemptive schedule whose value can be bounded in terms of the optimal LP value $Z_D = Z_R$. This yields results on the quality of both the computed schedule and the LP relaxations under consideration since the value of the computed schedule is an upper bound and the optimal LP value is a lower bound on the value of an optimal schedule.

In section 3.6, we describe a family of instances for which the ratio between the optimal value of the $1|r_j|\sum w_j C_j$ problem and the lower bounds Z_R and Z_D is arbitrarily close to $\frac{e}{e-1} > 1.5819$. This lower bound of $\frac{e}{e-1}$ sets a target for the design of approximation algorithms based on these LP relaxations.

In order to convert the preemptive LP schedule into a nonpreemptive schedule we make use of so-called α -points of jobs. For $0 < \alpha \leq 1$ the α -point $t_j(\alpha)$ of job j is the first point in time when an α -fraction of job j has been completed in the LP schedule, i.e., when j has been processed for αp_j time units. In particular, $t_j(1)$ is equal to the completion time and we define $t_j(0^+)$ to be the start time of job j . Notice that, by definition, the mean busy time M_j^{LP} of job j in the LP schedule is the average over all its α -points

$$(3.1) \quad M_j^{LP} = \int_0^1 t_j(\alpha) d\alpha.$$

We will also use the following notation: For a fixed job j and $0 < \alpha \leq 1$ we denote the fraction of job k that is completed in the LP schedule by time $t_j(\alpha)$ by $\eta_k(\alpha)$; in particular, $\eta_j(\alpha) = \alpha$. The amount of idle time that occurs between time 0 and the start of job j in the LP schedule is denoted by τ_{idle} . Note that η_k and τ_{idle} implicitly depend on the fixed job j . By construction, there is no idle time between the start and completion of job j in the LP schedule; therefore we can express j 's α -point as

$$(3.2) \quad t_j(\alpha) = \tau_{idle} + \sum_{k \in N} \eta_k(\alpha) p_k.$$

For a given $0 < \alpha \leq 1$, we define the α -schedule as the schedule in which jobs are processed nonpreemptively as early as possible and in the order of nondecreasing α -points. We denote the completion time of job j in this schedule by C_j^α . The idea of scheduling nonpreemptively in the order of α -points in a preemptive schedule was introduced by Phillips, Stein, and Wein [21] and used in many of the subsequent results in the area.

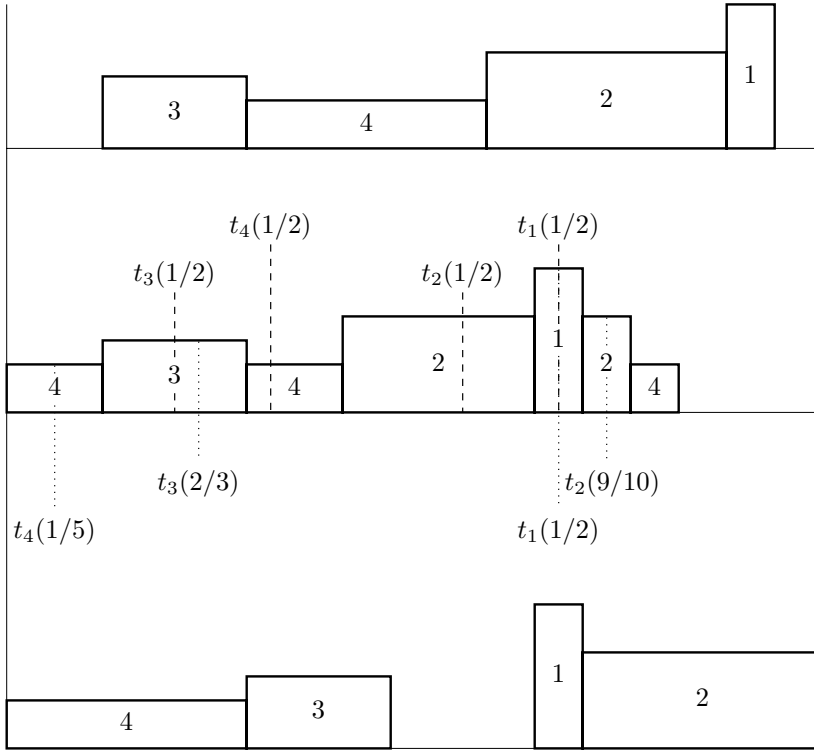


FIG. 2. A nonpreemptive α -schedule (for $\alpha = 1/2$) and an (α_j) -schedule shown above and below the LP schedule, respectively. Notice that there is no common α value that would lead to the latter schedule.

This idea can be further extended to individual, i.e., job-dependent α_j -points $t_j(\alpha_j)$, for $j \in N$ and $0 < \alpha_j \leq 1$. We denote the vector consisting of all α_j 's by $\boldsymbol{\alpha} := (\alpha_j) := (\alpha_1, \dots, \alpha_n)$. Then, the (α_j) -schedule is constructed by processing the jobs as early as possible and in nondecreasing order of their α_j -points; the completion time of job j in the (α_j) -schedule is denoted by $C_j^{\boldsymbol{\alpha}}$. Figure 2 compares an α -schedule to an (α_j) -schedule both derived from the LP schedule in Figure 1.

In what follows we present several results on the quality of α -schedules and (α_j) -schedules. These results also imply bounds on the quality of the LP relaxations discussed in the previous section. The main result is the construction of a random (α_j) -schedule whose expected value is at most a factor 1.6853 of the optimal LP value $Z_D = Z_R$. Therefore, the LP Relaxations (D) and (R) deliver a lower bound which is at least 0.5933 ($\approx 1.6853^{-1}$) times the optimal value. The corresponding randomized algorithm can be implemented on-line; it has competitive ratio 1.6853 and running time $O(n \log n)$; it can also be derandomized to run off-line in $O(n^2)$ time. We also investigate the case of a single common α and show that the best α -schedule is always within a factor of 1.7451 of the optimum.

3.1. Bounding the completion times in (α_j) -schedules. To analyze the completion times of jobs in (α_j) -schedules, we consider nonpreemptive schedules of similar structure that are, however, constructed by a slightly different conversion routine which we call (α_j) -CONVERSION.

Consider the jobs $j \in N$ in order of nonincreasing α_j -points $t_j(\alpha_j)$

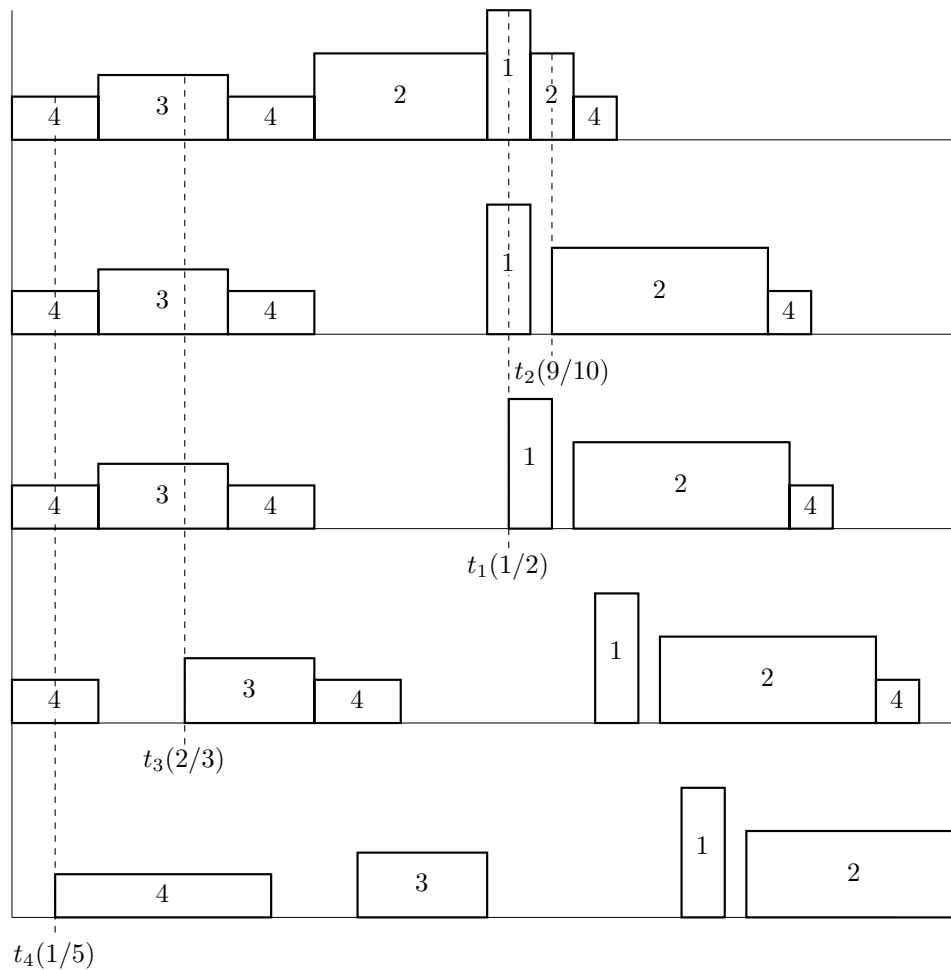


FIG. 3. Illustration of the individual iterations of (α_j) -CONVERSION.

and iteratively change the preemptive LP schedule to a nonpreemptive schedule by applying the following steps:

- (i) remove the $\alpha_j p_j$ units of job j that are processed before $t_j(\alpha_j)$ and leave the machine idle during the corresponding time intervals; we say that this idle time *is caused by* job j ;
- (ii) delay the whole processing that is done later than $t_j(\alpha_j)$ by p_j ;
- (iii) remove the remaining $(1 - \alpha_j)$ -fraction of job j from the machine and shrink the corresponding time intervals; *shrinking* a time interval means to discard the interval and move earlier, by the corresponding amount, any processing that occurs later;
- (iv) process job j in the released time interval $[t_j(\alpha_j), t_j(\alpha_j) + p_j]$.

Figure 3 contains an example illustrating the action of (α_j) -CONVERSION starting from the LP schedule of Figure 1. Observe that in the resulting schedule jobs are processed in nondecreasing order of α_j -points, and no job j is started before time $t_j(\alpha_j) \geq r_j$. The latter property will be useful in the analysis of on-line (α_j) -schedules.

LEMMA 3.1. *The completion time of job j in the schedule constructed by (α_j) -CONVERSION is equal to*

$$t_j(\alpha_j) + \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} (1 + \alpha_k - \eta_k(\alpha_j))p_k.$$

Proof. Consider the schedule constructed by (α_j) -CONVERSION. The completion time of job j is equal to the idle time before its start plus the sum of processing times of jobs that start no later than j . Since the jobs are processed in nondecreasing order of their α_j -points, the amount of processing before the completion of job j is

$$(3.3) \quad \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} p_k.$$

The idle time before the start of job j can be written as the sum of the idle time τ_{idle} that already existed in the LP schedule before j 's start plus the idle time before the start of job j that is caused in step (i) of (α_j) -CONVERSION; notice that step (iii) does not create any additional idle time since we shrink the affected time intervals. Each job k that is started no later than j , i.e., such that $\eta_k(\alpha_j) \geq \alpha_k$, contributes $\alpha_k p_k$ units of idle time; all other jobs k only contribute $\eta_k(\alpha_j) p_k$ units of idle time. As a result, the total idle time before the start of job j can be written as

$$(3.4) \quad \tau_{idle} + \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} \alpha_k p_k + \sum_{\substack{k \\ \alpha_k > \eta_k(\alpha_j)}} \eta_k(\alpha_j) p_k.$$

The completion time of job j in the schedule constructed by (α_j) -CONVERSION is equal to the sum of the expressions in (3.3) and (3.4); the result then follows from (3.2). \square

It follows from Lemma 3.1 that the completion time C_j of each job j in the nonpreemptive schedule constructed by (α_j) -CONVERSION satisfies $C_j \geq t_j(\alpha_j) + p_j \geq r_j + p_j$; hence is a feasible schedule. Since the (α_j) -schedule processes the jobs as early as possible and in the same order as the (α_j) -CONVERSION schedule, we obtain the following corollary.

COROLLARY 3.2. *The completion time of job j in an (α_j) -schedule can be bounded by*

$$C_j^\alpha \leq t_j(\alpha_j) + \sum_{\substack{k \\ \alpha_k \leq \eta_k(\alpha_j)}} (1 + \alpha_k - \eta_k(\alpha_j))p_k.$$

3.2. Bounds for α -schedules and (α_j) -schedules. We start with a result on the quality of the α -schedule for a fixed common value of α .

THEOREM 3.3. *For fixed α , (i) the value of the α -schedule is within a factor $\max\{1 + \frac{1}{\alpha}, 1 + 2\alpha\}$ of the optimal LP value; in particular, for $\alpha = 1/\sqrt{2}$ the bound is $1 + \sqrt{2}$. Simultaneously, (ii) the length of the α -schedule is within a factor of $1 + \alpha$ of the optimal makespan.*

Proof. While the proof of (ii) is an immediate consequence of (3.2) and Corollary 3.2, it follows from the proof of Theorem 2.5 that for (i) it is sufficient to prove that, for any canonical set S , we have

$$(3.5) \quad \sum_{j \in S} p_j C_j^\alpha \leq \max\left(1 + \frac{1}{\alpha}, 1 + 2\alpha\right) \left\{ p(S) \left(r_{\min}(S) + \frac{1}{2} p(S) \right) + \frac{1}{2} \sum_{j \in S} p_j^2 \right\}.$$

Indeed, using (2.5) and Lemma 2.4 it would then follow that

$$\begin{aligned}
\sum_{j \in N} w_j C_j^\alpha &= \sum_{i=1}^n \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{\ell=1}^{k(i)} \sum_{j \in S_\ell^i} p_j C_j^\alpha \\
&\leq \max \left(1 + \frac{1}{\alpha}, 1 + 2\alpha \right) \sum_{i=1}^n \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{\ell=1}^{k(i)} \sum_{j \in S_\ell^i} p_j \left(M_j^{LP} + \frac{p_j}{2} \right) \\
&= \max \left(1 + \frac{1}{\alpha}, 1 + 2\alpha \right) \sum_{j \in N} w_j \left(M_j^{LP} + \frac{p_j}{2} \right) \\
&= \max \left(1 + \frac{1}{\alpha}, 1 + 2\alpha \right) Z_R,
\end{aligned}$$

thus proving the result.

Now consider any canonical set S and let us assume that, after renumbering the jobs, $S = \{1, 2, \dots, \ell\}$ and $t_1(\alpha) < t_2(\alpha) < \dots < t_\ell(\alpha)$ (so the ordering is not necessarily anymore in nonincreasing order of w_j/p_j). Now fix any job $j \in S$. From Corollary 3.2, we derive that

$$(3.6) \quad C_j^\alpha \leq t_j(\alpha) + \sum_{k: \alpha \leq \eta_k} (1 + \alpha - \eta_k) p_k,$$

where $\eta_k := \eta_k(\alpha)$ represents the fraction of job k processed in the LP schedule before $t_j(\alpha)$. Let R denote the set of jobs k such that $t_k(\alpha) < r_{\min}(S)$ (and thus $\alpha \leq \eta_k$). Since S is a canonical set, the jobs in S are processed continuously in the LP schedule between $r_{\min}(S)$ and $r_{\min}(S) + p(S)$, and therefore every job k with $\alpha \leq \eta_k$ is either in S or in R . Observe that $\sum_{k \in R} \alpha p_k \leq r_{\min}(S)$ implies that $p(R) \leq \frac{1}{\alpha} r_{\min}(S)$. We can thus simplify (3.6) with

$$(3.7) \quad C_j^\alpha \leq t_j(\alpha) + \frac{1}{\alpha} r_{\min}(S) + \sum_{k=1}^j (1 + \alpha - \eta_k) p_k.$$

Since the jobs in S are scheduled with no gaps in $[r_{\min}(S), r_{\min}(S) + p(S)]$, we have that

$$(3.8) \quad t_j(\alpha) = r_{\min}(S) + \sum_{k \in S} \eta_k p_k \leq r_{\min}(S) + \sum_{k=1}^j \eta_k p_k + \sum_{k=j+1}^{\ell} \alpha p_k.$$

Combining (3.7) and (3.8), we derive that

$$C_j^\alpha \leq \left(1 + \frac{1}{\alpha} \right) r_{\min}(S) + \alpha p(S) + \sum_{k=1}^j p_k.$$

Multiplying by p_j and summing over S , we get

$$\begin{aligned} \sum_{j \in S} p_j C_j^\alpha &\leq \left(1 + \frac{1}{\alpha}\right) r_{\min}(S) p(S) + \alpha p(S)^2 + \sum_{j \in S} \sum_{k=1}^j p_j p_k \\ &= \left(1 + \frac{1}{\alpha}\right) r_{\min}(S) p(S) + \left(\frac{1}{2} + \alpha\right) p(S)^2 + \frac{1}{2} \sum_{j \in S} p_j^2, \end{aligned}$$

which implies (3.5). \square

In what follows we will compare the completion time C_j^α of every job j with its “completion time” $M_j^{LP} + \frac{1}{2}p_j$ in the LP schedule. However, for any fixed common value of α , there exist instances which show that this type of job-by-job analysis can give a bound no better than $1 + \sqrt{2} > 2.4142$. One can also show that, for any given value of α , there exist instances for which the objective function value of the α -schedule can be as bad as twice the LP lower bound.

In view of these results, it is advantageous to use several values of α , as it appears that no instance can be simultaneously bad for all choices of α . In fact, α -points develop their full power in combination with randomization, i.e., when a common α or even job-dependent α_j 's are chosen randomly from $(0, 1]$ according to an appropriate density function. This is also motivated by (3.1) which relates the expected α -point of a job under a uniform distribution of α to the LP variable M_j^{LP} . For random values α_j , we analyze the expected value of the resulting (α_j) -schedule and compare it to the optimal LP value. Notice that a bound on the expected value proves the existence of a vector $(\bar{\alpha}_j)$ such that the corresponding $(\bar{\alpha}_j)$ -schedule meets this bound. Moreover, for our results we can always compute such an $(\bar{\alpha}_j)$ in polynomial time by derandomizing our algorithms with standard methods; see Propositions 3.8 and 3.13.

Although the currently best known bounds can be achieved only for (α_j) -schedules with job-dependent α_j 's, we investigate α -schedules with a single common α as well. On the one hand, this helps to better understand the potential advantages of (α_j) -schedules; on the other hand, the randomized algorithm that relies on a single α admits a natural derandomization. In fact, we can easily compute an α -schedule of least objective function value over *all* α between 0 and 1; we refer to this schedule as the BEST- α -schedule. In Proposition 3.8 below, we will show that there are at most n different α -schedules. The BEST- α -schedule can be constructed in $O(n^2)$ time by evaluating all these different schedules.

As a warm-up exercise for the kind of analysis we use, we start by proving a bound of 2 on the expected worst-case performance ratio of uniformly generated (α_j) -schedules in the following theorem. This result will then be improved by using more intricate probability distributions and by taking advantage of additional insights into the structure of the LP schedule.

THEOREM 3.4. *Let the random variables α_j be pairwise independently and uniformly drawn from $(0, 1]$. Then, the expected value of the resulting (α_j) -schedule is within a factor 2 of the optimal LP value $Z_D = Z_R$.*

Proof. Remember that the optimal LP value is given by $\sum_j w_j (M_j^{LP} + \frac{1}{2}p_j)$. To get the claimed result, we prove that $E_U[C_j^\alpha] \leq 2(M_j^{LP} + \frac{1}{2}p_j)$ for all jobs $j \in N$, where $E_U[F(\alpha)]$ denotes the expectation of a function F of the random variable α when the latter is uniformly distributed. The overall performance follows from this job-by-job bound by linearity of expectations.

Consider an arbitrary, but fixed job $j \in N$. To analyze the expected completion time of j , we first keep α_j fixed and consider the conditional expectation $E_U[C_j^\alpha | \alpha_j]$. Since the random variables α_j and α_k are independent for each $k \neq j$, Corollary 3.2

and (3.2) yield

$$\begin{aligned} \mathbb{E}_U[C_j^\alpha | \alpha_j] &\leq t_j(\alpha_j) + \sum_{k \neq j} p_k \int_0^{\eta_k(\alpha_j)} (1 + \alpha_k - \eta_k(\alpha_j)) d\alpha_k + p_j \\ &= t_j(\alpha_j) + \sum_{k \neq j} \left(\eta_k(\alpha_j) - \frac{\eta_k(\alpha_j)^2}{2} \right) p_k + p_j \\ &\leq t_j(\alpha_j) + \sum_{k \neq j} \eta_k(\alpha_j) p_k + p_j \leq 2 \left(t_j(\alpha_j) + \frac{1}{2} p_j \right). \end{aligned}$$

To obtain the unconditional expectation $\mathbb{E}_U[C_j^\alpha]$ we integrate over all possible choices of α_j

$$\mathbb{E}_U[C_j^\alpha] = \int_0^1 \mathbb{E}_U[C_j^\alpha | \alpha_j] d\alpha_j \leq 2 \left(\int_0^1 t_j(\alpha_j) d\alpha_j + \frac{1}{2} p_j \right) = 2 \left(M_j^{LP} + \frac{1}{2} p_j \right);$$

the last equation follows from (3.1). \square

We turn now to deriving improved results. We start with an analysis of the structure of the LP schedule. Consider any job j and assume that, in the LP schedule, j is preempted at time s and its processing resumes at time $t > s$. Then all jobs which are processed between s and t have a smaller index; as a result, these jobs will be completely processed between times s and t . Thus, in the LP schedule, between the start time and the completion time of any job j , the machine is constantly busy, alternating between the processing of portions of j and the complete processing of groups of jobs with a smaller index. Conversely, any job preempted at the start time $t_j(0^+)$ of job j will have to wait at least until job j is complete before its processing can be resumed.

We capture this structure by partitioning, for a fixed job j , the set of jobs $N \setminus \{j\}$ into two subsets N_1 and N_2 : Let N_2 denote the set of all jobs that are processed between the start and completion of job j . All remaining jobs are put into subset N_1 . Notice that the function η_k is constant for jobs $k \in N_1$; to simplify notation we write $\eta_k := \eta_k(\alpha_j)$ for those jobs. For $k \in N_2$, let $0 < \mu_k < 1$ denote the fraction of job j that is processed before the start of job k ; the function η_k is then given by

$$\eta_k(\alpha_j) = \begin{cases} 0 & \text{if } \alpha_j \leq \mu_k, \\ 1 & \text{if } \alpha_j > \mu_k \end{cases} \quad \text{for } k \in N_2.$$

We can now rewrite (3.2) as

$$(3.9) \quad t_j(\alpha_j) = \tau_{\text{idle}} + \sum_{k \in N_1} \eta_k p_k + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j p_j = t_j(0^+) + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} p_k + \alpha_j p_j.$$

Plugging (3.9) into (3.1) yields

$$(3.10) \quad M_j^{LP} = t_j(0^+) + \sum_{k \in N_2} (1 - \mu_k) p_k + \frac{1}{2} p_j,$$

and Corollary 3.2 can be rewritten as

(3.11)

$$C_j^\alpha \leq t_j(0^+) + \sum_{\substack{k \in N_1 \\ \alpha_k \leq \eta_k}} (1 + \alpha_k - \eta_k) p_k + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} (1 + \alpha_k) p_k + (1 + \alpha_j) p_j,$$

where, for $k \in N_2$, we have used the fact that $\alpha_k \leq \eta_k(\alpha_j)$ is equivalent to $\alpha_j > \mu_k$. The expressions (3.9), (3.10), and (3.11) reflect the structural insights that we need for proving stronger bounds for (α_j) -schedules and α -schedules in what follows.

As mentioned above, the second ingredient for an improvement on the bound of 2 is a more sophisticated probability distribution of the random variables α_j . In view of the bound on C_j^α given in (3.11), we have to cope with two contrary phenomena: On the one hand, small values of α_k keep the terms of the form $(1 + \alpha_k - \eta_k)$ and $(1 + \alpha_k)$ on the right-hand side of (3.11) small; on the other hand, choosing larger values decreases the number of terms in the first sum on the right-hand side of (3.11). The balancing of these two effects contributes to reducing the bound on the expected value of C_j^α .

3.3. Improved bounds for α -schedules. In this subsection we prove the following theorem.

THEOREM 3.5. *Let $\gamma \approx 0.4675$ be the unique solution to the equation*

$$1 - \frac{\gamma^2}{1 + \gamma} = \gamma + \ln(1 + \gamma)$$

satisfying $0 < \gamma < 1$. Define $c := \frac{1+\gamma}{1+\gamma-e^{-\gamma}} < 1.7451$ and $\delta := 1 - \frac{\gamma^2}{1+\gamma} \approx 0.8511$. If α is chosen according to the density function

$$f(\alpha) = \begin{cases} (c - 1)e^\alpha & \text{if } \alpha \leq \delta, \\ 0 & \text{otherwise,} \end{cases}$$

then the expected value of the resulting random α -schedule is bounded by c times the optimal LP value $Z_D = Z_R$.

Before we prove Theorem 3.5 we state two properties of the density function f that are crucial for the analysis of the corresponding random α -schedule.

LEMMA 3.6. *The function f given in Theorem 3.5 is a density function with the following properties:*

- (i) $\int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha \leq (c - 1)\eta$ for all $\eta \in [0, 1]$,
- (ii) $\int_\mu^1 f(\alpha)(1 + \alpha) d\alpha \leq c(1 - \mu)$ for all $\mu \in [0, 1]$.

Property (i) is used to bound the delay of job j caused by jobs in N_1 , which corresponds to the first summation on the right-hand side of (3.11). The second summation reflects the delay of job j caused by jobs in N_2 and will be bounded by property (ii).

Proof of Lemma 3.6. A short computation shows that $\delta = \ln \frac{c}{c-1}$. The function f is a density function since

$$\int_0^1 f(\alpha) d\alpha = (c - 1) \int_0^\delta e^\alpha d\alpha = (c - 1) \left(\frac{c}{c - 1} - 1 \right) = 1.$$

In order to prove property (i), observe that for $\eta \in [0, \delta]$

$$\int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha = (c - 1) \int_0^\eta e^\alpha(1 + \alpha - \eta) d\alpha = (c - 1)\eta.$$

For $\eta \in (\delta, 1]$ we therefore get

$$\int_0^\eta f(\alpha)(1 + \alpha - \eta) d\alpha < \int_0^\delta f(\alpha)(1 + \alpha - \delta) d\alpha = (c - 1)\delta < (c - 1)\eta.$$

Property (ii) holds for $\mu \in (\delta, 1]$ since the left-hand side is 0 in this case. For $\mu \in [0, \delta]$ we have

$$\begin{aligned} \int_\mu^1 f(\alpha)(1 + \alpha) d\alpha &= (c - 1) \int_\mu^\delta e^\alpha(1 + \alpha) d\alpha = (c - 1)(e^\delta \delta - e^\mu \mu) \\ &= c \left(1 - \frac{\gamma^2}{1 + \gamma} - \frac{e^{\mu - \gamma} \mu}{1 + \gamma} \right) \leq c \left(1 - \frac{\gamma^2 + (1 + \mu - \gamma)\mu}{1 + \gamma} \right) \\ &= c \left(1 - \frac{(\gamma - \mu)^2 + (1 + \gamma)\mu}{1 + \gamma} \right) \leq c(1 - \mu). \end{aligned}$$

This completes the proof of the lemma. \square

Proof of Theorem 3.5. In Lemma 3.6, both (i) for $\eta = 1$ and (ii) for $\mu = 0$ yield $E_f[\alpha] \leq c - 1$, where $E_f[\alpha]$ denotes the expected value of a random variable α that is distributed according to the density function f given in Theorem 3.5. Thus, using inequality (3.11) and Lemma 3.6 we derive that

$$\begin{aligned} E_f[C_j^\alpha] &\leq t_j(0^+) + (c - 1) \sum_{k \in N_1} \eta_k p_k + c \sum_{k \in N_2} (1 - \mu_k) p_k + c p_j \\ &\leq c t_j(0^+) + c \sum_{k \in N_2} (1 - \mu_k) p_k + c p_j = c \left(M_j^{LP} + \frac{1}{2} p_j \right); \end{aligned}$$

the last inequality follows from the definition of N_1 and η_k , and the last equality follows from (3.10). \square

Notice that any density function satisfying properties (i) and (ii) of Lemma 3.6 for some value c' directly leads to the job-by-job bound $E_f[C_j^\alpha] \leq c' (M_j^{LP} + \frac{1}{2} p_j)$ for the corresponding random α -schedule. It is easy to see that the unit function satisfies Lemma 3.6 with $c' = 2$, which establishes the following variant of Theorem 3.4.

COROLLARY 3.7. *Let the random variable α be uniformly drawn from $(0, 1]$. Then, the expected value of the resulting α -schedule is within a factor 2 of the optimal LP value $Z_D = Z_R$.*

The use of an exponential density function is motivated by the first property in Lemma 3.6; notice that the function $\alpha \mapsto (c - 1)e^\alpha$ satisfies it with equality. On the other hand, the exponential function is truncated in order to reduce the term $\int_\mu^1 f(\alpha)(1 + \alpha) d\alpha$ in the second property. In fact, the truncated exponential function f in Theorem 3.5 can be shown to minimize c' ; it is therefore optimal for our analysis. In addition, there exists a class of instances for which the ratio of the expected cost of an α -schedule, determined using this density function, to the cost of the optimal LP value is arbitrarily close to 1.745; this shows that the preceding analysis is essentially tight in conjunction with truncated exponential functions.

Theorem 3.5 implies that the BEST- α -schedule has a value of at most $1.7451 Z_R$. The following proposition shows that the randomized algorithm that yields the α -schedule can be easily derandomized because the sample space is small.

PROPOSITION 3.8. *There are at most n different α -schedules; they can be computed in $O(n^2)$ time.*

Proof. As α goes from 0^+ to 1, the α -schedule changes only whenever an α -point, say for job j , reaches a time at which job j is preempted. Thus, the total number of changes in the α -schedule is bounded from above by the total number of preemptions. Since a preemption can occur in the LP schedule only whenever a job is released, the total number of preemptions is at most $n - 1$, and the number of α -schedules is at most n . Since each of these α -schedules can be computed in $O(n)$ time, the result on the running time follows. \square

3.4. Improved bounds for (α_j) -schedules. In this subsection, we prove the following theorem.

THEOREM 3.9. *Let $\gamma \approx 0.4835$ be the unique solution to the equation*

$$\gamma + \ln(2 - \gamma) = e^{-\gamma}((2 - \gamma)e^\gamma - 1)$$

satisfying $0 < \gamma < 1$. Define $\delta := \gamma + \ln(2 - \gamma) \approx 0.8999$ and $c := 1 + e^{-\gamma}/\delta < 1.6853$. Let the α_j 's be chosen pairwise independently from a probability distribution over $(0, 1]$ with the density function

$$g(\alpha) = \begin{cases} (c - 1)e^\alpha & \text{if } \alpha \leq \delta, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the expected value of the resulting random (α_j) -schedule is bounded by c times the optimal LP value $Z_D = Z_R$.

The bound in Theorem 3.9 also yields a bound on the quality of the LP relaxations.

COROLLARY 3.10. *The LP Relaxations (D) and (R) deliver in $O(n \log n)$ time a lower bound which is at least 0.5933 ($\approx 1.6853^{-1}$) times the objective function value of an optimal schedule.*

Following the lines of the last subsection, we state two properties of the density function g that are crucial for the analysis of the corresponding random (α_j) -schedule.

LEMMA 3.11. *The function g given in Theorem 3.9 is a density function with the following properties:*

- (i) $\int_0^\eta g(\alpha)(1 + \alpha - \eta) d\alpha \leq (c - 1)\eta$ for all $\eta \in [0, 1]$,
- (ii) $(1 + E_g[\alpha]) \int_\mu^1 g(\alpha) d\alpha \leq c(1 - \mu)$ for all $\mu \in [0, 1]$,

where $E_g[\alpha]$ denotes the expected value of a random variable α that is distributed according to g .

Notice the similarity of Lemma 3.11 and Lemma 3.6 of the last subsection. Again, properties (i) and (ii) are used to bound the delay of job j caused by jobs in N_1 and N_2 , respectively, in the right-hand side of inequality (3.11). Property (i) for $\eta = 1$ or property (ii) for $\mu = 0$ again yield $E_g[\alpha] \leq c - 1$.

Proof of Lemma 3.11. A short computation shows that $\delta = \ln \frac{c}{c-1}$. It thus follows from the same arguments as in the proof of Lemma 3.6 that g is a density function and that property (i) holds. In order to prove property (ii), we first compute

$$E_g[\alpha] = \int_0^1 g(\alpha)\alpha d\alpha = (c - 1) \int_0^\delta e^\alpha \alpha d\alpha = c\delta - 1.$$

Property (ii) certainly holds for $\mu \in (\delta, 1]$. For $\mu \in [0, \delta]$ we get

$$\begin{aligned} (1 + \mathbb{E}_g[\alpha]) \int_{\mu}^1 g(\alpha) d\alpha &= c\delta(c-1) \int_{\mu}^{\delta} e^{\alpha} d\alpha \\ &= ce^{-\gamma}((2-\gamma)e^{\gamma} - e^{\mu}) \\ &= c(2-\gamma - e^{\mu-\gamma}) \\ &\leq c(2-\gamma - (1+\mu-\gamma)) \\ &= c(1-\mu). \end{aligned}$$

This completes the proof of the lemma. \square

Proof of Theorem 3.9. Our analysis of the expected completion time of job j in the random (α_j) -schedule follows the line of argument developed in the proof of Theorem 3.4. First we consider a fixed choice of α_j and bound the corresponding conditional expectation $\mathbb{E}_g[C_j^{\alpha} | \alpha_j]$. In a second step we bound the unconditional expectation $\mathbb{E}_g[C_j^{\alpha}]$ by integrating the product $g(\alpha_j)\mathbb{E}_g[C_j^{\alpha} | \alpha_j]$ over the interval $(0, 1]$.

For a fixed job j and a fixed value α_j , the bound in (3.11) and Lemma 3.11 (i) yield

$$\begin{aligned} \mathbb{E}_g[C_j^{\alpha} | \alpha_j] &\leq t_j(0^+) + (c-1) \sum_{k \in N_1} \eta_k p_k + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} (1 + \mathbb{E}_g[\alpha_k])p_k + (1 + \alpha_j)p_j \\ &\leq ct_j(0^+) + (1 + \mathbb{E}_g[\alpha_1]) \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} p_k + (1 + \alpha_j)p_j. \end{aligned}$$

The last inequality follows from (3.9) and $\mathbb{E}_g[\alpha_k] = \mathbb{E}_g[\alpha_1]$ for all $k \in N$. Using property (ii) and (3.10) yields

$$\begin{aligned} \mathbb{E}_g[C_j^{\alpha}] &\leq ct_j(0^+) + (1 + \mathbb{E}_g[\alpha_1]) \sum_{k \in N_2} p_k \int_{\mu_k}^1 g(\alpha_j) d\alpha_j + (1 + \mathbb{E}_g[\alpha_j])p_j \\ &\leq ct_j(0^+) + c \sum_{k \in N_2} (1 - \mu_k)p_k + cp_j = c \left(M_j^{LP} + \frac{1}{2}p_j \right). \end{aligned}$$

The result follows from linearity of expectations. \square

While the total number of possible orderings of jobs is $n! = 2^{O(n \log n)}$, we show in the following lemma that the maximal number of (α_j) -schedules is at most 2^{n-1} . We will use the following observation. Let q_j denote the number of different pieces of job j in the LP schedule; thus q_j represents the number of times job j is preempted plus 1. Since there are at most $n-1$ preemptions, we have that $\sum_{j=1}^n q_j \leq 2n-1$.

LEMMA 3.12. *The maximal number of (α_j) -schedules is at most 2^{n-1} , and this bound can be attained.*

Proof. The number of (α_j) -schedules is given by $s = \prod_{j=1}^n q_j$. Notice that $q_1 = 1$ since this job is not preempted in the LP schedule. Thus, $s = \prod_{j=2}^n q_j$, while $\sum_{j=2}^n q_j \leq 2(n-1)$. By the arithmetic-geometric mean inequality, we have that

$$s = \prod_{j=2}^n q_j \leq \left(\frac{\sum_{j=2}^n q_j}{n-1} \right)^{n-1} \leq 2^{n-1}.$$

Furthermore, this bound is attained if $q_j = 2$ for $j = 2, \dots, n$, and this is achieved, for example, for the instance with $p_j = 2$, $w_j = n - j + 1$, and $r_j = n - j$ for all j . \square

Therefore, and in contrast to the case of random α -schedules, we cannot afford to derandomize the randomized 1.6853-approximation algorithm by enumerating all (α_j) -schedules. We instead use the method of conditional probabilities [18].

From inequality (3.11) we obtain for every vector $\alpha = (\alpha_j)$ an upper bound on the objective function value of the corresponding (α_j) -schedule, $\sum_{j \in N} w_j C_j^\alpha \leq UB(\alpha)$, where $UB(\alpha) = \sum_{j \in N} w_j RHS_j(\alpha)$ and $RHS_j(\alpha)$ denotes the right-hand side of inequality (3.11). Taking expectations and using Theorem 3.9, we have already shown that

$$E_g \left[\sum_{j \in N} w_j C_j^\alpha \right] \leq E_g[UB(\alpha)] \leq c Z_D,$$

where $c < 1.6853$. For each job $j \in N$ let $\mathcal{Q}_j = \{Q_{j1}, \dots, Q_{jq_j}\}$ denote the set of intervals for α_j corresponding to the q_j pieces of job j in the LP schedule. We consider the jobs one by one in arbitrary order, say, $j = 1, \dots, n$. Assume that, at step j of the derandomized algorithm, we have identified intervals $Q_1^d \in \mathcal{Q}_1, \dots, Q_{j-1}^d \in \mathcal{Q}_{j-1}$ such that

$$E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j-1] \leq c Z_D.$$

Using conditional expectations, the left-hand side of this inequality is

$$\begin{aligned} & E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j-1] \\ &= \sum_{\ell=1}^{q_j} \text{Prob}\{\alpha_j \in Q_{j\ell}\} E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j-1 \text{ and } \alpha_j \in Q_{j\ell}]. \end{aligned}$$

Since $\sum_{\ell=1}^{q_j} \text{Prob}\{\alpha_j \in Q_{j\ell}\} = 1$, there exists at least one interval $Q_{j\ell} \in \mathcal{Q}_j$ such that

$$(3.12) \quad \begin{aligned} & E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j-1 \text{ and } \alpha_j \in Q_{j\ell}] \\ & \leq E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j-1]. \end{aligned}$$

Therefore, it suffices to identify such an interval $Q_j^d = Q_{j\ell}$ satisfying (3.12), and we may conclude that

$$\begin{aligned} & E_g \left[\sum_{h \in N} w_h C_h^\alpha \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j \right] \\ & \leq E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, j] \leq c Z_D. \end{aligned}$$

Having determined in this way an interval Q_j^d for every job $j = 1, \dots, n$, we then note that the (α_j) -schedule is the same for all $\alpha \in Q_1^d \times Q_2^d \times \dots \times Q_n^d$. The (now deterministic) objective function value of this (α_j) -schedule is

$$\begin{aligned} \sum_{j \in N} w_j C_j^\alpha & \leq E_g[UB(\alpha) \mid \alpha_i \in Q_i^d \text{ for } i = 1, \dots, n] \\ & \leq E_g[UB(\alpha)] \leq c Z_D < 1.6853 Z_D, \end{aligned}$$

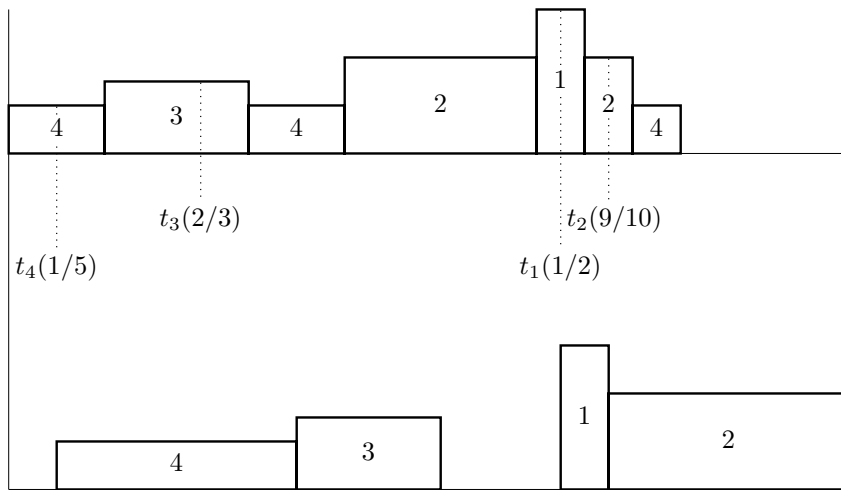


FIG. 4. The on-line schedule for the previously considered instance and α_j -points. The LP schedule is shown above for comparison.

as desired. For every $j = 1, \dots, n$, checking whether an interval Q_j^d satisfies inequality (3.12) amounts to evaluating $O(n)$ terms, each of which may be computed in constant time. Since, as observed just before Lemma 3.12, we have a total of $\sum_{j=1}^n q_j \leq 2n - 1$ candidate intervals, it follows that the derandomized algorithm runs in $O(n^2)$ time.

PROPOSITION 3.13. *The randomized 1.6853-approximation algorithm can be derandomized; the resulting deterministic algorithm runs in $O(n^2)$ time and has performance guarantee 1.6853 as well.*

3.5. Constructing provably good schedules on-line. In this subsection we show that our randomized approximation results also apply in an on-line setting. There are several different on-line paradigms that have been studied in the area of scheduling; we refer to [30] for a survey. We consider the setting where jobs continually arrive over time, and, for each time t , we must construct the schedule until time t without any knowledge of the jobs that will arrive afterwards. In particular, the characteristics of a job, i.e., its processing time and its weight, become known only at its release date.

It has already been shown in section 2 that the LP schedule can be constructed on-line. Unfortunately, for a given vector (α_j) , the corresponding (α_j) -schedule cannot be constructed on-line. We learn only about the position of a job k in the sequence defined by nondecreasing α_j -points at time $t_k(\alpha_k)$; therefore, we cannot start job k at an earlier point in time in the on-line setting. On the other hand, however, the start time of k in the (α_j) -schedule can be earlier than its α_k -point $t_k(\alpha_k)$.

Although an (α_j) -schedule cannot be constructed on-line, the above discussion reveals that the following variant, which we call the *on-line- (α_j) -schedule*, can be constructed on-line: For a given vector (α_j) , process the jobs as early as possible in the order of their α_j -points, with the additional constraint that no job k may start before time $t_k(\alpha_k)$. See Figure 4 for an example. We note that this idea of delaying the start of jobs until sufficient information for a good decision is available was in this setting introduced by Phillips, Stein, and Wein [21].

Notice that the nonpreemptive schedule constructed by (α_j) -CONVERSION fulfills these constraints; its value is therefore an upper bound on the value of the on-line- (α_j) -schedule. Our analysis in the last subsections relies on the bound given in Corollary 3.2, which also holds for the schedule constructed by (α_j) -CONVERSION by Lemma 3.1. This yields the following results.

THEOREM 3.14. *For any instance of the scheduling problem $1|r_j|\sum w_j C_j$,*

- (a) *choosing $\alpha = 1/\sqrt{2}$ and constructing the on-line- α -schedule yields a deterministic on-line algorithm with competitive ratio $1 + \sqrt{2} \leq 2.4143$ and running time $O(n \log n)$;*
- (b) *choosing the α_j 's randomly and pairwise independently from $(0, 1]$ according to the density function g of Theorem 3.9 and constructing the on-line- (α_j) -schedule yields a randomized on-line algorithm with competitive ratio 1.6853 and running time $O(n \log n)$.*

The competitive ratio 1.6853 in Theorem 3.14 beats the deterministic on-line lower bound 2 for the unit-weight problem $1|r_j|\sum C_j$ [15, 36]. For the same problem, Stougie and Vestjens [37] (see also [39]) proved the lower bound $\frac{e}{e-1} > 1.5819$ for randomized on-line algorithms.

3.6. Bad instances for the LP relaxations. In this subsection, we describe a family of instances for which the ratio between the optimal value of the $1|r_j|\sum w_j C_j$ problem and the lower bounds Z_R and Z_D is arbitrarily close to $\frac{e}{e-1} > 1.5819$.

These instances I_n have $n \geq 2$ jobs as follows: one large job, denoted job n , and $n - 1$ small jobs, denoted $j = 1, \dots, n - 1$. The large job has processing time $p_n = n$, weight $w_n = \frac{1}{n}$, and release date $r_n = 0$. Each of the $n - 1$ small jobs j has zero processing time, weight $w_j = \frac{1}{n(n-1)}(1 + \frac{1}{n-1})^{n-j}$, and release date $r_j = j$.

Throughout the paper, we have assumed that processing times are nonzero. In order to satisfy this assumption, we could impose a processing time of $1/k$ for all small jobs, multiply all processing times and release dates by k to make the data integral, and then let k tend to infinity. For simplicity, however, we just let the processing time of all small jobs be 0.

The LP solution has job n start at time 0, preempted by each of the small jobs; hence its mean busy times are $M_j^{LP} = r_j$ for $j = 1, \dots, n - 1$ and $M_n^{LP} = \frac{n}{2}$. Its objective function value is $Z_R = (1 + \frac{1}{n-1})^n - (1 + \frac{1}{n-1})$. Notice that the completion time of each job j is in fact equal to $M_j^{LP} + \frac{1}{2}p_j$ such that the actual value of the preemptive schedule is equal to Z_R .

Now consider an optimal nonpreemptive schedule C^* and let $k = \lfloor C_n^* \rfloor - n \geq 0$, so k is the number of small jobs that can be processed before job n . It is then optimal to process all these small jobs $1, \dots, k$ at their release dates and to start processing job n at date $r_k = k$ just after job k . It is also optimal to process all remaining jobs $k + 1, \dots, n - 1$ at date $k + n$ just after job n . Let C^k denote the resulting schedule; that is, $C_j^k = j$ for all $j \leq k$, and $C_j^k = k + n$ otherwise. Its objective function value is $(1 + \frac{1}{n-1})^n - \frac{1}{n-1} - \frac{k}{n(n-1)}$. Therefore, the optimal schedule is C^{n-1} with objective function value $(1 + \frac{1}{n-1})^n - \frac{1}{n-1} - \frac{1}{n}$. As n grows large, the LP objective function value approaches $e - 1$ while the optimal nonpreemptive cost approaches e .

4. Conclusion. Even though polynomial-time approximation schemes have now been discovered for the problem $1|r_j|\sum w_j C_j$ [1], the algorithms we have developed, or variants of them, are likely to be superior in practice. The experimental studies of Savelsbergh, Uma, and Wein [25] and Uma and Wein [38] indicate that LP-based

relaxations and scheduling in order of α_j -points are powerful tools for a variety of scheduling problems.

Several intriguing questions remain open. Regarding the quality of linear programming relaxations, it would be interesting to close the gap between the upper (1.6853) and lower (1.5819) bound on the quality of the relaxations considered in this paper. We should point out that the situation for the strongly NP-hard [16] problem $1|r_j, pmtn|\sum w_j C_j$ is similar. It is shown in [29] that the completion time relaxation is in the worst case at least a factor of $8/7$ and at most a factor of $4/3$ off the optimum; the latter bound is achieved by scheduling preemptively by LP-based random α -points. Chekuri et al. [6] prove that the optimal nonpreemptive value is at most $e/(e-1)$ times the optimal preemptive value; our example in section 3.6 shows that this bound is tight.

Dyer and Wolsey [9] also propose a (nonpreemptive) time-indexed relaxation which is stronger than the preemptive version studied here. This relaxation involves variables for each job and each time representing whether this job is being *completed* (rather than simply processed) at that time. This relaxation is at least as strong as the preemptive version, but its worst-case ratio is not known to be strictly better.

For randomized on-line algorithms, there is also a gap between the known upper and lower bound on the competitive ratios that are given at the end of section 3.5. For deterministic on-line algorithms, the 2-competitive algorithm of Anderson and Potts [2] is optimal.

Acknowledgment. The authors are grateful to an anonymous referee whose comments helped to improve the presentation of this paper.

REFERENCES

- [1] F. AFRATI, E. BAMPIS, C. CHEKURI, D. KARGER, C. KENYON, S. KHANNA, I. MILIS, M. QUEYRANNE, M. SKUTELLA, C. STEIN, AND M. SVIRIDENKO, *Approximation schemes for minimizing average weighted completion time with release dates*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, New York, NY, 1999, pp. 32–43.
- [2] E. J. ANDERSON AND C. N. POTTS, *On-line scheduling of a single machine to minimize total weighted completion time*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 2002, pp. 548–557.
- [3] E. BALAS, *On the facial structure of scheduling polyhedra*, Math. Programming Stud., 24 (1985), pp. 179–218.
- [4] H. BELOUADAH, M. E. POSNER, AND C. N. POTTS, *Scheduling with release dates on a single machine to minimize total weighted completion time*, Discrete Appl. Math., 36 (1992), pp. 213–231.
- [5] S. CHAKRABARTI, C. PHILLIPS, A. S. SCHULZ, D. B. SHMOYS, C. STEIN, AND J. WEIN, *Improved scheduling algorithms for minsum criteria*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1099, F. Meyer auf der Heide and B. Monien, eds., Springer, Berlin, 1996, pp. 646–657.
- [6] C. CHEKURI, R. MOTWANI, B. NATARAJAN, AND C. STEIN, *Approximation techniques for average completion time scheduling*, SIAM J. Comput., 31 (2001), pp. 146–166.
- [7] C.-F. CHOU, M. QUEYRANNE, AND D. SIMCHI-LEVI, *The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 2081, K. Aardal and B. Gerards, eds., Springer, Berlin, 2001, pp. 45–59.
- [8] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.
- [9] M. E. DYER AND L. A. WOLSEY, *Formulating the single machine sequencing problem with release dates as a mixed integer program*, Discrete Appl. Math., 26 (1990), pp. 255–270.
- [10] M. X. GOEMANS, *A supermodular relaxation for scheduling with release dates*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H.

- Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, 1996, pp. 288–300.
- [11] M. X. GOEMANS, *Improved approximation algorithms for scheduling with release dates*, in Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, ACM, New York, 1997, pp. 591–598.
- [12] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Ann. Discrete Math., 5 (1979), pp. 287–326.
- [13] L. A. HALL, A. S. SCHULZ, D. B. SHMOYS, AND J. WEIN, *Scheduling to minimize average completion time: Off-line and on-line approximation algorithms*, Math. Oper. Res., 22 (1997), pp. 513–544.
- [14] L. A. HALL, D. B. SHMOYS, AND J. WEIN, *Scheduling to minimize average completion time: Off-line and on-line algorithms*, in Proceedings of the 7th Annual ACM–SIAM Symposium on Discrete Algorithms, Atlanta, GA, 1996, ACM, New York, 1996, pp. 142–151.
- [15] J. A. HOOGEVEEN AND A. P. A. VESTJENS, *Optimal on-line algorithms for single-machine scheduling*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, 1996, pp. 404–414.
- [16] J. LABETOULLE, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Preemptive scheduling of uniform machines subject to release dates*, in Progress in Combinatorial Optimization, W. R. Pulleyblank, ed., Academic Press, New York, 1984, pp. 245–261.
- [17] J. K. LENSTRA, A. H. G. RINNOOY KAN, AND P. BRUCKER, *Complexity of machine scheduling problems*, Ann. Discrete Math., 1 (1977), pp. 343–362.
- [18] R. MOTWANI AND P. RAGHAVAN, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.
- [19] A. MUNIER, M. QUEYRANNE, AND A. S. SCHULZ, *Approximation bounds for a general class of precedence constrained parallel machine scheduling problems*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Springer, Berlin, 1998, pp. 367–382, SIAM J. Comput., to appear.
- [20] G. L. NEMHAUSER, L. A. WOLSEY, AND M. L. FISHER, *An analysis of approximations for maximizing submodular set functions. I*, Math. Programming, 14 (1978), pp. 265–294.
- [21] C. PHILLIPS, C. STEIN, AND J. WEIN, *Minimizing average completion time in the presence of release dates*, Math. Programming, 82 (1998), pp. 199–223.
- [22] M. QUEYRANNE, *Structure of a simple scheduling polyhedron*, Math. Programming, 58 (1993), pp. 263–285.
- [23] M. QUEYRANNE AND A. S. SCHULZ, *Polyhedral Approaches to Machine Scheduling*, Preprint 408, Department of Mathematics, Technische Universität Berlin, Germany, 1994.
- [24] M. QUEYRANNE AND A. S. SCHULZ, *Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 920, Springer, Berlin, 1995, pp. 307–320.
- [25] M. W. P. SAVELSBERGH, R. N. UMA, AND J. M. WEIN, *An experimental study of LP-based approximation algorithms for scheduling problems*, in Proceedings of the 9th Annual ACM–SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, ACM, New York, 1998, pp. 453–462.
- [26] A. S. SCHULZ, *Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, 1996, pp. 301–315.
- [27] A. S. SCHULZ AND M. SKUTELLA, *Random-based scheduling: New approximations and LP lower bounds*, in Randomization and Approximation Techniques in Computer Science, Lecture Notes in Comput. Sci. 1269, J. Rolim, ed., Springer, Berlin, 1997, pp. 119–133.
- [28] A. S. SCHULZ AND M. SKUTELLA, *Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria*, in Algorithms—ESA '97, Lecture Notes in Comput. Sci. 1284, Springer, Berlin, 1997, pp. 416–429, SIAM J. Discrete Math., to appear under the title “Scheduling unrelated machines by randomized rounding.”
- [29] A. S. SCHULZ AND M. SKUTELLA, *The power of α -points in preemptive single machine scheduling*, J. Sched., 5 (2002), to appear.
- [30] J. SGALL, *On-line scheduling—a survey*, in Online Algorithms: The State of the Art, Lecture Notes in Comput. Sci. 1441, A. Fiat and G. J. Woeginger, eds., Springer, Berlin, 1998, pp. 196–231.
- [31] L. S. SHAPLEY, *Cores of convex games*, Internat. J. Game Theory, 1 (1971), pp. 11–26.

- [32] M. SKUTELLA, *Approximation and Randomization in Scheduling*, Ph.D. thesis, Technische Universität Berlin, Germany, 1998.
- [33] M. SKUTELLA, *Convex quadratic and semidefinite programming relaxations in scheduling*, J. ACM, 48 (2001), pp. 206–242.
- [34] M. SKUTELLA AND G. J. WOEGERING, *A PTAS for minimizing the total weighted completion time on identical parallel machines*, Math. Oper. Res., 25 (2000), pp. 63–75.
- [35] W. E. SMITH, *Various optimizers for single-stage production*, Naval Res. Logist. Quart., 3 (1956), pp. 59–66.
- [36] L. STOUGIE, *private communication* in [15], 1995.
- [37] L. STOUGIE AND A. P. A. VESTJENS, *Randomized algorithms for on-line scheduling problems: How low can't you go?*, Oper. Res. Lett., to appear.
- [38] R. N. UMA AND J. M. WEIN, *On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Springer, Berlin, 1998, pp. 394–408.
- [39] A. P. A. VESTJENS, *On-Line Machine Scheduling*, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1997.

FINDING A 2-CORE OF A TREE IN LINEAR TIME*

BIING-FENG WANG[†]

Abstract. Let T be an edge-weighted tree. A p -core of T is a set of p mutually disjoint paths in T that minimizes the sum of the distances of all vertices in T from any of the p paths, where $p \geq 1$ is an integer. In this paper, an $O(n)$ time algorithm is proposed for the case $p = 2$, where n is the number of vertices in T . Our algorithm improves the two $O(n^2)$ time algorithms previously proposed by Becker and Perl [*Discrete Appl. Math.*, 11 (1985), pp. 103–113]. With some modifications, the proposed algorithm can be implemented on the EREW PRAM in $O(\log^2 n)$ time using $O(n \log n)$ work.

Key words. graphs, algorithms, trees, cores, network location theory, parallel algorithms

AMS subject classifications. 05C85, 68W10

PII. S0895480100374242

1. Introduction. Network location theory is concerned with the optimal locations of service facilities in a network. The shapes of the facilities can be points, paths, or trees. A variety of network location problems have been defined and studied in the literature [2, 3, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29]. These location problems usually have important applications in transportation and communication and thus have received much attention from researchers in the fields.

Let T be an edge-weighted tree of n vertices. Let $p \geq 1$ be an integer. A p -median of T is a set of p vertices in T that minimizes the sum, over all vertices v of T , of the minimum distance of v to any of the p vertices. Goldman [7] gave a linear time algorithm for finding a 1-median of a tree. Gavish and Sridhar [6] presented an $O(n \log n)$ time algorithm for finding a 2-median of a tree. Using an algorithm based on dynamic programming, Tamir [26] solved the general p -median problem on a tree in $O(pn^2)$ time. A p -center of T is a set of p vertices in T that minimizes the distance to the furthest vertex from any of the p vertices. Linear time algorithms for finding a 1-center and a 2-center of a tree had been proposed by Handler and Mirchandani [10]. By developing an elegant tree decomposition scheme to find the k th longest path in a tree, Megiddo et al. [13] solved the general p -center problem on a tree in $O(n \log^2 n)$ time. Later, Frederickson and Johnson [5] improved the upper bound to $O(n \log n)$.

A p -core of T is a set of p mutually disjoint paths in T that minimizes the sum of the distances of all vertices in T from any of the p paths. The case $p = 1$ of the problem was first studied by Slater [23]. A linear time algorithm was proposed by Morgan and Slater [16]. A cost-optimal parallel algorithm that requires $O(\log n)$ time on the EREW PRAM was proposed by Peng and Lo [18]. Minięka and Patel [15] studied the problem of finding in a tree a 1-core of a specified length, which is a variant of the 1-core problem. Minięka [14] gave an $O(n^3)$ time algorithm for the

*Received by the editors June 20, 2000; accepted for publication (in revised form) January 2, 2002; published electronically March 20, 2002. This research was supported by the National Science Council of the Republic of China under grant NSC-90-2213-E-007-061. A preliminary version of this paper was presented at the 11th Annual International Symposium on Algorithms and Computation, Taipei, Taiwan, 2000.

<http://www.siam.org/journals/sidma/15-2/37424.html>

[†]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China (bfbwang@cs.nthu.edu.tw).

problem. Peng and Lo [20] gave an $O(n \log n)$ time algorithm for unweighted trees. Later, Alstrup et al. [2] presented an $O(n)$ time algorithm for unweighted trees and an $O(n \log n \alpha(n))$ time algorithm for weighted trees. Parallel algorithms on the EREW PRAM for finding a core of a specified length were proposed in [20, 28].

The 2-core problem was first considered by Becker and Perl [3]. They proposed two efficient algorithms for the problem. The first requires $O(n^2)$ time and the second requires $O(dn)$ time, where d is the maximum number of edges of any simple path in T . In the worst case and average case, the second algorithm requires $O(n^2)$ time and $O(n\sqrt{n})$ time, respectively. Novik [17] studied the general p -core problem and gave an $O(pn^2)$ time algorithm. Hakimi, Schmeichel, and Labbe [9] generalized Minieka and Patel's consideration to study the problem of finding a p -core of a specified length, which is called *the p -path median problem*. They showed that the problem is *NP*-hard if p is a variable and gave an $O(n^{2p+1})$ time algorithm for any constant p .

In this paper, an $O(n)$ time algorithm is proposed for finding a 2-core of a tree. The proposed algorithm is based upon several important properties explored by Becker and Perl [3] and thus can be regarded as a fast implementation of their algorithms. With some modification, the proposed algorithm can be implemented on the EREW PRAM in $O(\log^2 n)$ time using $O(n \log n)$ work. On the CRCW PRAM, the parallel running time can be further improved to $O(\log \log n \log n)$.

The remainder of this paper is organized as follows. In the next section, notation and preliminary results are presented. In section 3, a linear time algorithm is proposed to solve an optimization problem, which is called the r -point core problem. In section 4, using the algorithm proposed in section 3 as a key procedure, an $O(n)$ time algorithm for computing a 2-core of a tree is presented. In section 5, we give a parallel implementation of our 2-core algorithm on the EREW PRAM. Finally, in section 6, we conclude this paper.

2. Notation and preliminary results. In this section, we introduce notation and definitions that are used throughout this paper and give some preliminary results.

Let $T = (V, E)$ be a free tree. Let $n = |V|$. Each edge $e \in E$ has an associated positive length $w(e)$. If $w(e) = 1$ for every $e \in E$, then T is unweighted; otherwise T is weighted. For any two vertices $u, v \in V$, let $P(u, v)$ be the unique path from u to v and let $d(u, v)$ be its length. A path in T is a v -path, where $v \in V$, if v is one of its endpoints. For each $v \in V$, let $N(v)$ be the set of vertices in T adjacent to v . Consider a given vertex $v \in V$. The are $|N(v)|$ subtrees of T attached to v through the edges incident on v . For each $u \in N(v)$, denote by T_u^v the subtree of T attached to v through the edge (u, v) , excluding this edge and the vertex v .

For a subgraph X of T , the vertex set and edge set of X is $V(X)$ and $E(X)$, respectively. For easy description, given a vertex $v \in V$, the subgraph having vertex set $\{v\}$ and edge set \emptyset is simply denoted by v . Given two subgraphs X and Y of T , we denote by $X \cup Y$ the subgraph having vertex set $V(X) \cup V(Y)$ and edge set $E(X) \cup E(Y)$. For a vertex $v \in V$ and a subgraph X of T , the *distance* from v to X is $d(v, X) = \min_{u \in V(X)} d(v, u)$ and *close* (v, X) is the vertex in X nearest to v . For two subgraphs Y and X of T , the *distancesum* from Y to X is $D(Y, X) = \sum_{v \in V(Y)} d(v, X)$. If $Y = T$, we simply write $D(X)$ in place of $D(T, X)$.

A *1-core* of T is a path A in T that minimizes $D(A)$. A p -core of T is a set of p mutually disjoint paths $\{A_1, A_2, \dots, A_p\}$ in T that minimizes $D(A_1 \cup A_2 \cup \dots \cup A_p)$, where $p \geq 2$ is an integer.

Our 2-core algorithm in section 4 needs to solve two variants of the 1-core problem, which are defined as follows. Let $r \in V$ be a given vertex. An r *rooted-1-core* of T is

an r -path A in T that minimizes $D(A)$. An r -point core of T is a path A in T that does not contain r and minimizes $D(r \cup A)$.

The *distance saving* of a path $P(x, y)$ in T is $\delta(P(x, y)) = D(x) - D(P(x, y))$, where $x, y \in V$. For each edge $(u, v) \in E$, let $n_u^v = |V(T_u^v)|$, $s_u^v = D(T_u^v, v)$, l_u^v be a vertex in T_u^v having $\delta(P(v, l_u^v)) = \max_{x \in V(T_u^v)} \delta(P(v, x))$ and let $m_u^v = \delta(P(v, l_u^v))$.

In the remainder of this section, some preliminary results are given.

Morgan and Slater [16] proposed the following two results.

LEMMA 1 (see [16]). *The values of n_u^v , s_u^v , l_u^v , and m_u^v can be computed in $O(n)$ time for every $(u, v) \in E$.*

LEMMA 2 (see [16]). *A 1-core of a tree can be computed in $O(n)$ time.*

For each $v \in V$, $D(v) = \sum_{u \in N(v)} D(T_u^v, v) = \sum_{u \in N(v)} s_u^v$. Thus, we can obtain the following result from Lemma 1 easily.

LEMMA 3. *The value of $D(v)$ can be computed in $O(n)$ time for every $v \in V$.*

Peng, Stephen, and Yesha gave the following result.

LEMMA 4 (see [19]). *An r rooted-1-core of a tree can be computed in $O(n)$ time.*

The following lemma shows that the concept of distance saving is very useful for computing the distancesum from T to a given subgraph.

LEMMA 5. *Let Y be a subgraph of T . Let $(s, t) \in E$ be an edge such that $s \in V(Y)$ and $V(T_t^s) \cap V(Y) = \emptyset$. Let x be a vertex in T_t^s . Then, $D(Y \cup P(s, x)) = D(Y) - \delta(P(s, x))$.*

Proof. See Figure 1. Since $V(T_t^s) \cap V(Y) = \emptyset$, we have $D(Y \cup P(s, x)) = D(T_s^t, Y) + D(T_t^s, P(s, x))$. Clearly, $D(T_s^t, Y) = D(Y) - D(T_s^t, s)$ and $D(T_t^s, P(s, x)) = D(P(s, x)) - D(T_s^t, s)$. Thus, we have

$$\begin{aligned} D(Y \cup P(s, x)) &= D(Y) - D(T_s^t, s) - D(T_t^s, s) + D(P(s, x)) \\ &= D(Y) - D(s) + D(P(s, x)). \end{aligned}$$

By definition, $D(s) - D(P(s, x)) = \delta(P(s, x))$. Therefore, $D(Y \cup P(s, x)) = D(Y) - \delta(P(s, x))$ and the lemma holds. \square

3. Finding an r -point core in linear time. Let $r \in V$ be a vertex. In this section, an $O(n)$ time algorithm is proposed for finding an r -point core of T . It is applied as a key procedure in our 2-core algorithm in the next section.

Assuming that T is unweighted, Becker and Perl [3] had proposed an $O(n)$ time algorithm for the r -point core problem. Their algorithm can be applied to a weighted tree. However, $O(n \log n)$ time is required. In this section, we show that an r -point core of a weighted tree T can be found in $O(n)$ time. Our r -point core algorithm is obtained by modifying Becker and Perl's algorithm. Therefore, we begin by describing their algorithm.

A *bisector* of a simple path (v_1, v_2, \dots, v_m) is an edge (v_k, v_{k+1}) such that $d(v_1, v_k) \leq d(v_1, v_m)/2 \leq d(v_1, v_{k+1})$, where $1 \leq k < m$. (It will be *the* bisector if there does not exist a vertex v_c such that $d(v_1, v_c) = d(v_1, v_m)/2$.)

LEMMA 6 (see [3]). *Let $a, b \in V$. Let (x, y) be an edge in E such that x is nearer to a than to b and (x, y) is a bisector of $P(a, b)$. We have $D(a \cup b) = D(a) - (s_y^x + n_y^x \times d(a, x)) + D(b) - (s_x^y + n_x^y \times d(b, y))$.*

Proof. See Figure 2. Since (x, y) is a bisector of $P(a, b)$, $D(a \cup b) = D(T_x^y, a) + D(T_y^x, b)$. We have

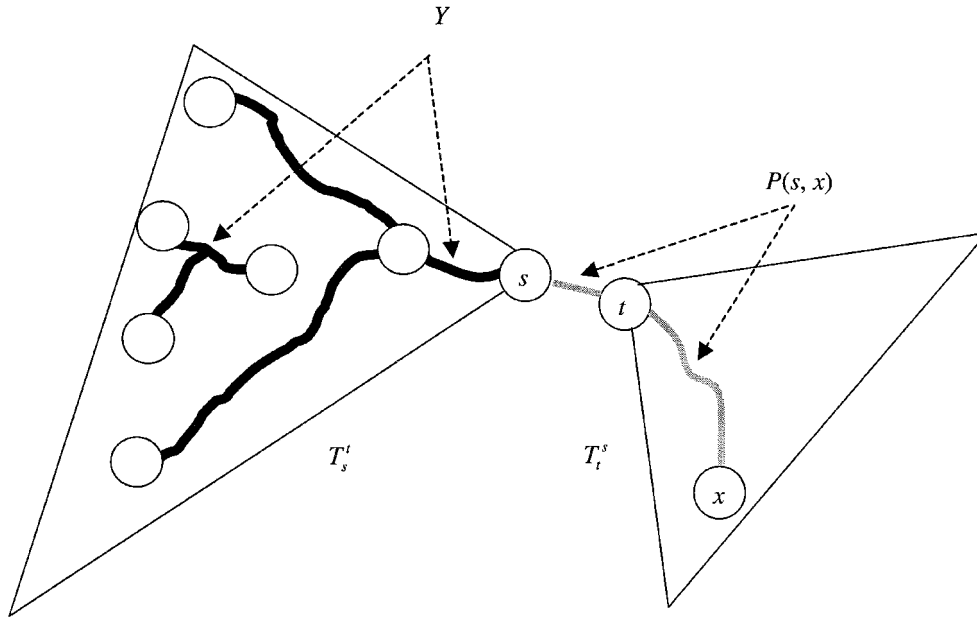


FIG. 1. An illustration for the proof of Lemma 5.

$$\begin{aligned}
 D(T_x^y, a) &= D(a) - D(T_y^x, a) \\
 &= D(a) - (D(T_y^x, x) + |T_y^x| \times d(a, x)) \\
 &\quad (\text{since } d(v, a) = d(v, x) + d(x, a) \text{ for every } v \in V(T_y^x)) \\
 &= D(a) - (s_y^x + n_y^x \times d(a, x)).
 \end{aligned}$$

Similarly, we have $D(T_y^x, b) = D(b) - (s_x^y + n_x^y \times d(b, y))$. Therefore, the lemma holds. \square

For easy description, in the remainder of this section, we assume that T is a rooted tree with root r .

LEMMA 7 (see [3]). *Let $v \in V$ be a vertex such that there is an r -point core A of T satisfying $\text{close}(r, A) = v$. Let u_1 and u_2 be the two sons of v maximizing $m_{u_1}^v + m_{u_2}^v$. Then, $P(l_{u_1}^v, l_{u_2}^v)$ is an r -point core of T and $D(P(l_{u_1}^v, l_{u_2}^v)) = D(r \cup v) - (m_{u_1}^v + m_{u_2}^v)$.*

Based upon Lemmas 6 and 7, Becker and Perl [3] presented the following algorithm for finding an r -point core.

ALGORITHM 1. POINT_CORE(T, r)

Input: a tree $T = (V, E)$ and a vertex $r \in V$

Output: an r -point core of T

begin

1. Orient T into a rooted tree with root r
2. **for** each $(u, v) \in E$ **do** compute n_u^v, s_u^v, l_u^v , and m_u^v
3. **for** each $v \in V$ **do** compute $D(v)$ and $d(r, v)$
4. $W \leftarrow V - \{r\}$ /* W contains at least one $v \in V$ satisfying the condition in Lemma 7. */
5. **for** each $v \in W$ **do** $(x_v, y_v) \leftarrow$ a bisector of $P(r, v)$ such that x_v is the parent of y_v

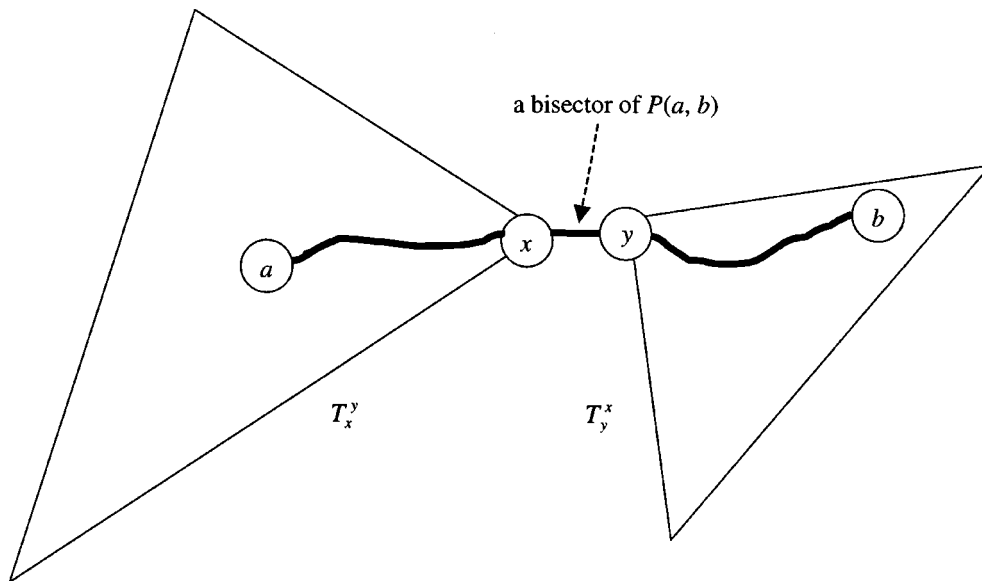


FIG. 2. An illustration for the proof of Lemma 6.

6. **for** each $v \in W$ **do**
 7. $(u_1, u_2) \leftarrow$ the two sons of v maximizing $m_{u_1}^v + m_{u_2}^v$
 8. $A_v \leftarrow P(l_{u_1}^v, l_{u_2}^v)$
 9. $D(r \cup v) \leftarrow D(r) - (s_{y_v}^{x_v} + n_{y_v}^{x_v} \times d(r, x_v)) + D(v) - (s_{x_v}^{y_v} + n_{x_v}^{y_v} \times (d(r, v) - d(r, y_v)))$ /* Lemma 6*/
 10. $D(r \cup A_v) \leftarrow D(r \cup v) - (m_{u_1}^v + m_{u_2}^v)$ /* Lemma 7*/
 11. $CORE \leftarrow$ the A_z with $D(r \cup A_z) = \min_{v \in W} D(r \cup A_v)$
 12. **return**($CORE$)
- end.**

It is not difficult to check that except line 5, all computation in Algorithm 1 requires $O(n)$ time no matter whether T is unweighted or not. Becker and Perl assumed that T is unweighted and implemented line 5 in linear time by performing a depth-first traversal on T maintaining the path from the root r to the current vertex v in an array H such that $H[i]$ stores the i th vertex on $P(r, v)$. While a vertex v is visited during the traversal, (x_v, y_v) is computed in $O(1)$ time as $(H[\lfloor l/2 \rfloor], H[\lfloor l/2 \rfloor + 1])$, where $l = d(r, v)$. We have the following theorem.

THEOREM 1 (see [3]). *Finding an r -point core of an unweighted tree can be done in $O(n)$ time.*

Line 5 of Algorithm 1 computes for every $v \in V - \{r\}$ a bisector of $P(r, v)$. Wang, Ku, and Shi [29] showed that the computation has an $\Omega(n \log n)$ time lower bound if T is weighted. Thus, applying Becker and Perl's algorithm to a weighted tree requires $\Omega(n \log n)$ time. The lower bound can be easily achieved as follows. We perform a depth-first traversal on T maintaining two arrays H and D such that $H[i]$ stores the i th vertex on $P(r, v)$ and $D[i]$ stores $d(r, H[i])$. During the traversal, a bisector edge of $P(r, v)$ is determined in $O(\log n)$ time by performing a binary search for $l/2$ on D , where $l = d(r, v)$.

THEOREM 2. *Finding an r -point core of a weighted tree can be done in $O(n \log n)$ time.*

In the remainder of this section, we show that finding an r -point core of a weighted tree T can be done in $O(n)$ time by giving some modifications to Algorithm 1.

LEMMA 8. *Let $k \in V$ and $P(k, g)$ be a k rooted-1-core of T . For any $v \in V$, we have $\delta(P(c, g)) \geq \delta(P(c, v))$, where $c = \text{close}(v, P(k, g))$.*

Proof. By letting $Y = P(k, c)$, $s = c$, and $x = g$ in Lemma 5, we have $D(P(k, g)) = D(P(k, c)) - \delta(P(c, g))$. Similarly, by letting $Y = P(k, c)$, $s = c$, and $x = v$ in Lemma 5, we have $D(P(k, v)) = D(P(k, c)) - \delta(P(c, v))$. Since $P(k, g)$ is an r rooted-1-core, $D(P(k, g)) \leq D(P(k, v))$ and thus $\delta(P(c, g)) \geq \delta(P(c, v))$. Therefore, the lemma holds. \square

Recall that T was assumed to be a rooted tree with root r . Every $u \in N(r)$ is a son of r . For each $u \in N(r)$, we denote by X_u the subtree of T rooted at u . Let A be an r -point core of T . Since A does not contain r , there is a vertex $k \in N(r)$ such that A is totally contained in X_k . Let M_k be a k rooted-1-core of X_k . We have the following lemma.

LEMMA 9. $V(A) \cap V(M_k) \neq \emptyset$.

Proof. See Figure 3. We prove this lemma by contradiction. Suppose that $V(A) \cap V(M_k) = \emptyset$. Let $A = P(w, z)$ and $a = \text{close}(k, A)$. Let $M_k = P(k, g)$ and $b = \text{close}(a, M_k)$. Let $A' = P(g, z)$, which is $P(g, b) \cup P(b, a) \cup P(a, z)$. By using Lemma 5 twice, we can obtain $D(r \cup A) = D(r \cup a) - \delta(P(a, w)) - \delta(P(a, z))$. Again, by using Lemma 5 twice, we can obtain $D(r \cup A') = D(r \cup P(b, a)) - \delta(P(b, g)) - \delta(P(a, z))$. Thus, $D(r \cup A') - D(r \cup A) = D(r \cup P(b, a)) - D(r \cup a) - \delta(P(b, g)) + \delta(P(a, w))$. Since M_k is a k rooted-1-core of X_k , by Lemma 8, $\delta(P(b, g)) \geq \delta(P(b, w)) \geq \delta(P(a, w))$. Therefore, $D(r \cup A') - D(r \cup A) \leq D(r \cup P(b, a)) - D(r \cup a)$. Since $V(A) \cap V(M_k) = \emptyset$, we have $a \neq b$. Thus, $r \cup a$ is a proper subgraph of $r \cup P(b, a)$. Therefore, $D(r \cup P(b, a)) < D(r \cup a)$. Consequently, $D(r \cup A') - D(r \cup A) < 0$, which contradicts the fact that A is an r -point core of T . Therefore, $V(A) \cap V(M_k) \neq \emptyset$. \square

LEMMA 10. $\text{close}(r, A) \in V(M_k)$.

Proof. By Lemma 9, $V(A) \cap V(M_k) \neq \emptyset$. Let a be the vertex in $V(A) \cap V(M_k)$ that is nearest to r . Let a' be the parent of a . Since $a' \notin V(A)$, $a \in V(A)$, and A is a path, A is totally contained in the subtree rooted at a . Therefore, $a = \text{close}(r, A)$ and the lemma holds. \square

For each $u \in N(r)$, let M_u be a u rooted-1-core of X_u . According to Lemma 10, $\text{close}(r, A) \in \bigcup_{u \in N(r)} V(M_u)$ for any r -point core A of T . Therefore, we can modify Algorithm 1 by replacing line 4 with the following.

4. $W \leftarrow \bigcup_{u \in N(r)} V(M_u)$, where M_u is a u rooted-1-core of T_u^r .

In the following, we show that after the replacement, Algorithm 1 can be implemented in $O(n)$ time no matter whether T is unweighted or not. By Lemma 4, the computation of all M_u , where $u \in N(r)$, requires $O(\sum_{u \in N(r)} |V(T_u^r)|) = O(n)$ time. Thus, line 4 takes $O(n)$ time. Consider a fixed vertex $u \in N(r)$. Since M_u is a path, we can easily compute a bisector of $P(r, v)$ for every $v \in V(M_u)$ in $O(|V(M_u)|)$ time by resorting to a linear time merging algorithm. Thus, line 5 can be implemented in $O(\sum_{u \in N(r)} |V(M_u)|) = O(n)$ time. All the other computation of Algorithm 1 requires $O(n)$ time. Therefore, we obtain the following theorem.

THEOREM 3. *An r -point core of a weighted tree can be computed in $O(n)$ time.*

4. Finding a 2-core in linear time. In this section, we propose an $O(n)$ time algorithm for finding a 2-core of T . For easy description, we define a (a, b) rooted-2-core of T , where $a, b \in V$ and $a \neq b$, as a pair of two disjoint paths (A, B) such that

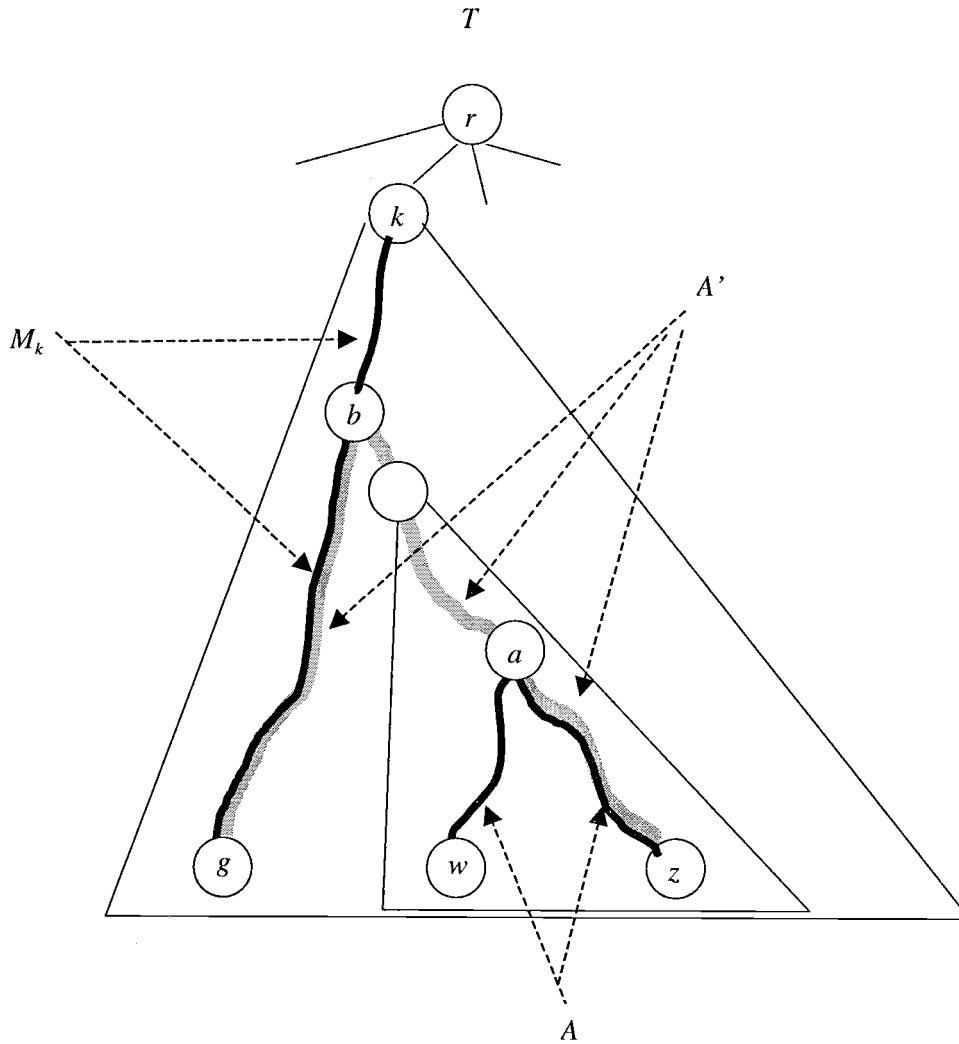


FIG. 3. An illustration for the proof of Lemma 9.

A is an a -path, B is a b -path, and $D(A \cup B)$ is minimized. One of the most critical steps of our 2-core algorithm is to compute a (a, b) rooted-2-core of T for two given vertices $a, b \in V$. The computation is complicated. We describe it first in subsection 4.1. Our 2-core algorithm is then proposed in subsection 4.2.

4.1. Finding a rooted-2-core. Let $a, b \in V$ be two vertices. Becker and Perl [3] had proposed an $O(n^2)$ time algorithm for the problem of finding a (a, b) rooted-2-core of T . In this subsection, we give an $O(n)$ time algorithm for the same problem.

We begin by giving some notations. Let (c_1, c_2, \dots, c_d) be the sequence of vertices on $P(a, b)$, where $c_1 = a$ and $c_d = b$. For each $i, 1 \leq i < d$, let Q_i be the subtree $T_{c_i}^{c_{i+1}}$ and U_i be the subtree $T_{c_{i+1}}^{c_i}$. (See Figure 4.) Let S_1, S_2, \dots, S_d be the subtrees obtained from T by deleting all edges on $P(a, b)$ such that $S_i, 1 \leq i \leq d$, is the one containing c_i .

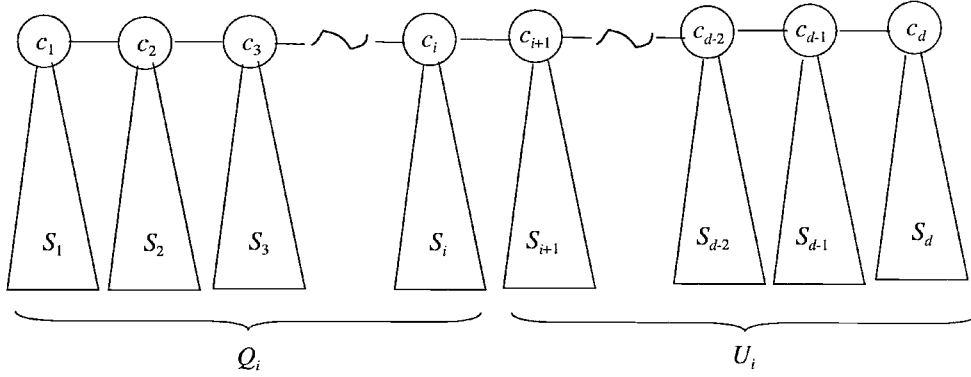


FIG. 4. Subtrees S_i , Q_i , and U_i .

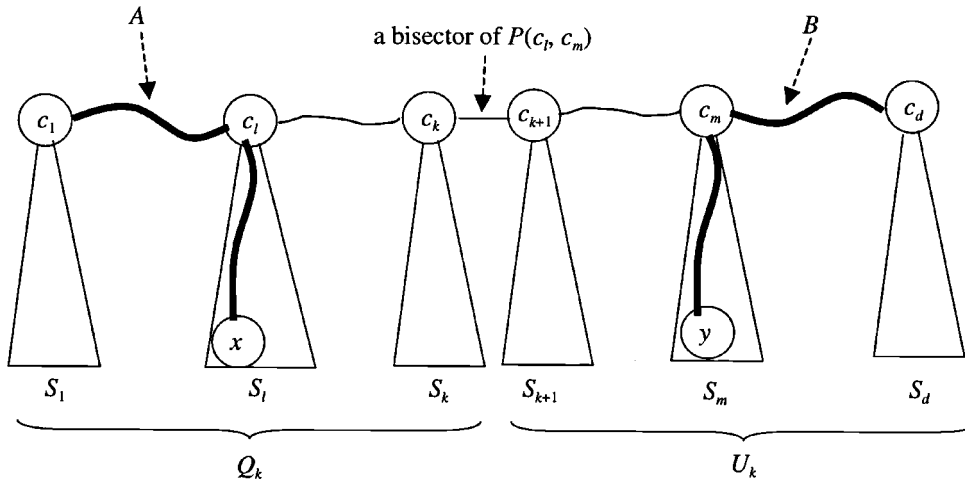


FIG. 5. A (a, b) rooted-2-core of T , where $c_1 = a$ and $c_d = b$.

Let $A = P(a, x)$ and $B = P(b, y)$ be two paths such that (A, B) is a (a, b) rooted-2-core of T . Let S_l and S_m , respectively, be the two subtrees containing x and y , where $1 \leq l \leq d$ and $1 \leq m \leq d$. (See Figure 5.) Since A and B are disjoint, we have $l < m$, c_l is the vertex on A that is nearest to B , and c_m is the vertex on B that is nearest to A . Let (c_k, c_{k+1}) be a bisector of $P(c_l, c_m)$, where $l \leq k < m$. According to the definition of bisectors, vertices in Q_k are nearer to A than to B ; and vertices in U_k are nearer to B than to A . Thus, $D(A \cup B) = D(Q_k, A) + D(U_k, B)$. Since (A, B) is a (a, b) rooted-2-core of T , we conclude that A is an a rooted-1-core of Q_k and B is a b rooted-1-core of U_k . Otherwise, there is another pair of disjoint paths $(P(a, x'), P(b, y'))$ having $D(P(a, x') \cup P(b, y')) < D(A \cup B)$, which contradicts the fact that (A, B) is a (a, b) rooted-2-core of T .

From the above discussion, we have the following lemma.

LEMMA 11. *There is an integer k , $1 \leq k < d$, such that an a rooted-1-core of Q_k and a b rooted-1-core of U_k constitute a (a, b) rooted-2-core of T .*

Therefore, we can compute a (a, b) rooted-2-core of T as follows.

ALGORITHM 2. ROOTED_2_CORE(T, a, b)

Input: a tree $T = (V, E)$ and two vertices $a, b \in V$

Output: a (a, b) rooted-2-core of T

begin

1. **for** each $k, 1 \leq k < d$, **do**
2. $A_k \leftarrow$ an a rooted-1-core of Q_k
3. $DA_k \rightarrow D(Q_k, A_k)$
4. **for** each $k, 1 \leq k < d$, **do**
5. $B_k \leftarrow$ a b rooted-1-core of U_k
6. $DB_k \leftarrow D(U_k, B_k)$
7. $CORE \leftarrow$ the (A_z, B_z) with $(DA_z + DB_z) = \min_{1 \leq k < d} (DA_k + DB_k)$
8. **return**($CORE$)

end.

As we will show later, lines 1–3 of Algorithm 2 can be done in $O(n)$ time. The computation of lines 4–6 is the same with that of lines 1–3 and thus can also be done in $O(n)$ time. Line 7 requires $O(d)$ time. Therefore, Algorithm 2 performs in $O(n)$ time. We have the following theorem.

THEOREM 4. *A rooted-2-core of a weighted tree can be computed in $O(n)$ time.*

To complete the proof of Theorem 4, in the following, we show that lines 1–3 of Algorithm 2 can be done in $O(n)$ time. That is, we show that finding an a rooted-1-core A_k for every $Q_k, 1 \leq k < d$, can be done in $O(n)$ time.

First, we describe a procedure to preprocess T . In the preprocessing, a vertex $g_i \in V(S_i)$ is computed for each subtree $S_i, 1 \leq i < d$, such that $P(c_i, g_i)$ is a c_i rooted-1-core of S_i . Besides, four auxiliary arrays L, UN, UD , and GD are computed such that $L[i] = d(a, c_i)$, $UN[i] = |V(U_i)|$, $UD[i] = D(U_i, c_i)$, and $GD[i] = D(P(a, g_i))$, where $1 \leq i < d$. The procedure is as follows.

PROCEDURE PREPROCESS($T, (c_1 = a, c_2, \dots, c_d = b)$)

begin

1. **for** each $i, 1 \leq i < d$, **do**
2. $g_i \leftarrow$ a vertex in S_i such that $P(c_i, g_i)$ is a c_i rooted-1-core of S_i
3. **for** each $i, 1 \leq i < d$, **do** $L[i] \leftarrow d(a, c_i)$
4. **for** each $i, 1 \leq i < d$, **do** $UN[i] \leftarrow n_{c_{i+1}}^{c_i}$ /* by definition, $|V(U_i)| = n_{c_{i+1}}^{c_i}$ */
5. **for** each $i, 1 \leq i < d$, **do** $UD[i] \leftarrow s_{c_{i+1}}^{c_i}$ /* by definition, $D(U_i, c_i) = s_{c_{i+1}}^{c_i}$ */
6. **for** each $i, 1 \leq i < d$, **do** $\delta(P(c_i, g_i)) \leftarrow D(S_i, c_i) - D(S_i, P(c_i, g_i))$
7. $D(P(a, c_1)) \leftarrow D(a)$
8. **for** each $i, 2 \leq i < d$, **do** $D(P(a, c_i)) \leftarrow D(P(a, c_{i-1})) - d(c_{i-1}, c_i)UN[i-1]$
9. **for** each $i, 1 \leq i < d$, **do** $GD[i] \leftarrow D(P(a, c_i)) - \delta(P(c_i, g_i))$ /* by Lemma 5 */
10. **return** ($g_1, g_2, \dots, g_{d-1}, L, UN, UD, GD$)

end.

By Lemma 4, lines 1–2 take $O(\sum_{1 \leq i < d} |V(S_i)|) = O(n)$ time. Trivially, line 3 takes $O(d)$ time. By Lemma 1, the computation of UN and UD in lines 4–5 takes $O(n)$ time. It is easy to check that $D(c_i) - D(P(c_i, g_i)) = D(S_i, c_i) - D(S_i, P(c_i, g_i))$, which validates the computation of $\delta(P(c_i, g_i))$ in line 6. The computation in line 6 takes $O(\sum_{1 \leq i < d} |V(S_i)|) = O(n)$ time. Clearly, lines 7–8 take $O(n)$ time. The reason that we can compute in line 8 $D(P(a, c_i))$ as $D(P(a, c_{i-1})) - d(c_{i-1}, c_i)UN[i-1]$, $2 \leq i < d$, is as follows. The two paths $P(a, c_i)$ and $P(a, c_{i-1})$ differ only in the edge

(c_{i-1}, c_i) . Thus, for each $v \in V(Q_{i-1})$, $d(v, P(a, c_i)) = d(v, P(a, c_{i-1}))$ and for each $v \in V(U_{i-1})$, $d(v, P(a, c_i)) = d(v, P(a, c_{i-1})) - d(c_{i-1}, c_i)$. Therefore, we conclude that $D(P(a, c_i)) = D(P(a, c_{i-1})) - d(c_{i-1}, c_i)|V(U_{i-1})|$. Thus, the computation in line 8 is valid. By letting $Y = P(a, c_i)$, $s = c_i$, and $x = g_i$ in Lemma 5, we obtain $D(P(a, g_i)) = D(P(a, c_i)) - \delta(P(c_i, g_i))$. Thus, line 9 correctly computes the array GD in $O(d)$ time. Therefore, the above preprocessing procedure is correct and requires $O(n)$ time in total.

For easy discussion, in the remainder of this subsection, k is assumed to be a fixed integer between 1 to $d - 1$ unless explicitly specified otherwise. Consider the computation of A_k . One of the two endpoints of A_k is a . To compute A_k , we shall select a correct vertex among the vertices in Q_k to be the other endpoint. Q_k consists of the subtrees S_i , $i = 1, 2, \dots, k$. The following lemma shows that in order to select a correct vertex to be the other endpoint of A_k , in each subtree S_i , $1 \leq i \leq k$, we can consider only the vertex g_i .

LEMMA 12. $D(Q_k, P(a, g_i)) = \min_{u \in V(S_i)} D(Q_k, P(a, u))$, $1 \leq i \leq k < d$.

Proof. Clearly, for every $u \in V(S_i)$, $D(Q_k, P(a, u)) = \sum_{1 \leq j < i} D(S_j, c_j) + D(S_i, P(c_i, u)) + \sum_{i < j \leq k} D(S_j, c_j)$. Thus, to minimize $D(Q_k, P(a, u))$ over all vertices $u \in V(S_i)$ is equivalent to minimizing $D(S_i, P(c_i, u))$. Since $P(c_i, g_i)$ is a c_i rooted-1-core of S_i , $D(S_i, P(c_i, g_i)) = \min_{u \in V(S_i)} D(S_i, P(c_i, u))$. Therefore, $D(Q_k, P(a, g_i)) = \min_{u \in V(S_i)} D(Q_k, P(a, u))$ \square

Let $G_i = P(a, g_i)$, $1 \leq i < d$. We call (G_1, G_2, \dots, G_k) the *candidate sequence* of A_k , since it can be easily concluded from Lemma 12 that the path in the sequence minimizing $D(Q_k, G_i)$ is an a rooted-1-core of Q_k .

LEMMA 13. $D(Q_k, G_i) = D(G_i) - D(U_k, c_k) - d(c_k, c_i)|V(U_k)|$, $1 \leq i \leq k < d$.

Proof. Since $V(Q_k)$ and $V(U_k)$ is a partition of V , we have $D(Q_k, G_i) = D(G_i) - D(U_k, G_i)$. For every vertex $v \in V(U_k)$, $d(v, G_i) = d(v, c_i) = d(v, c_k) + d(c_k, c_i)$. Thus, $D(U_k, G_i) = D(U_k, c_k) + d(c_k, c_i)|V(U_k)|$. And thus $D(Q_k, G_i) = D(G_i) - D(U_k, c_k) - d(c_k, c_i)|V(U_k)|$. Therefore, the lemma holds. \square

By Lemma 13, we can compute $D(Q_k, G_i) = GD[i] - UD[k] - (L[k] - L[i])UN[k]$ for any two given integers i and k , $1 \leq i \leq k < d$, in $O(1)$ time using the auxiliary arrays. Thus, we can determine A_k from its candidate sequence in $O(k)$ time. Consequently, all A_k , $k = 1, 2, \dots, d - 1$, can be computed in $O(1 + 2 + \dots + d - 1) = O(d^2)$ time. It is known that the average diameter of an unweighted tree is $O(\sqrt{n})$ [12]. Thus, $O(d^2) = O(n)$ in the average case. However in the worst case, $O(d^2) = O(n^2)$. We need more techniques to speed up the computation of A_k .

We define the function $f(G_x, G_y)$, for two paths G_x and G_y , $1 \leq x < y < d$, to be

$$f(G_x, G_y) = \frac{D(G_x) - D(G_y)}{d(c_x, c_y)}.$$

Using arrays L and GD , the value of $f(G_x, G_y)$ can be easily determined in $O(1)$ time for any two given integers x and y , $1 \leq x < y < d$.

LEMMA 14. For $1 \leq x < y \leq s < d$, to determine whether $D(Q_s, G_x) <, =, \text{ or } > D(Q_s, G_y)$, respectively, is equivalent to determining whether $f(G_x, G_y) <, =, \text{ or } > |V(U_s)|$.

Proof. By Lemma 13, we have $D(Q_s, G_x) - D(Q_s, G_y) = D(G_x) - D(G_y) - d(c_x, c_y)|V(U_s)|$, from which we can easily conclude that to determine whether $D(Q_s, G_x) <, =, \text{ or } > D(Q_s, G_y)$ is equivalent to determining whether $(D(G_x) - D(G_y))/d(c_x, c_y) <, =, \text{ or } > |V(U_s)|$. Therefore, the lemma holds. \square

Based upon Lemma 14, in the following, two properties are proposed to further recognize in (G_1, G_2, \dots, G_k) some paths that we can ignore while computing A_k .

Property 1. For $1 \leq x < y < k$, if $f(G_x, G_y) < f(G_y, G_k)$, then G_y cannot be an a rooted-1-core of $Q_k, Q_{k+1}, \dots, Q_{d-1}$.

Proof. Let Q_s be any of $Q_k, Q_{k+1}, \dots, Q_{d-1}$. We consider two cases: $f(G_x, G_y) < |V(U_s)|$ and $f(G_x, G_y) \geq |V(U_s)|$. If $f(G_x, G_y) < |V(U_s)|$, then by Lemma 14 $D(Q_s, G_x) < D(Q_s, G_y)$ and thus G_y cannot be an a rooted-1-core of Q_s . Assume that $f(G_x, G_y) \geq |V(U_s)|$. Since $f(G_y, G_k) > f(G_x, G_y) \geq |V(U_s)|$, by Lemma 14 $D(Q_s, G_y) > D(Q_s, G_k)$ and thus G_y cannot be an a rooted-1-core of Q_s . Therefore, the lemma holds. \square

Property 2. For $1 \leq x < y \leq k$, if $f(G_x, G_y) > |V(U_k)|$, then G_x cannot be an a rooted-1-core of $Q_k, Q_{k+1}, \dots, Q_{d-1}$.

Proof. Let Q_s be any of $Q_k, Q_{k+1}, \dots, Q_{d-1}$. Since $f(G_x, G_y) > |V(U_k)| \geq |V(U_s)|$, we obtain from Lemma 14 that $D(Q_s, G_x) > D(Q_s, G_y)$. Therefore, G_x cannot be an a rooted-1-core of Q_s and the lemma holds. \square

Now we are ready to present our algorithm for computing all $A_k, k = 1, 2, \dots, d-1$. Before describing the details, it is outlined as follows. In the algorithm, we compute the $d-1$ rooted-1-cores, one at a time, from A_1 to A_{d-1} . To save time, during the computation, we maintain a sequence D of paths. Initially, D is an empty sequence. Before the computation of A_k , D is adjusted by adding G_k and removing according to Properties 1 and 2 some paths that we can ignore while computing A_k, A_{k+1}, \dots , and A_{d-1} . After the adjustment, we then compute A_k as the G_i in D that minimizes $D(Q_k, G_i)$.

We begin to describe the details. The path sequence D is implemented as a *deque* [12], which is defined as a doubly linked list allowing $O(1)$ time insertion and deletion at both ends. We denote by $|D|$, *Head*, and *Tail*, respectively, the number of paths, the first path, and the last path in D . And, for any path w in D , we denote by $w.next$ and $w.last$, respectively, the next and last paths of w in D . Initially, D is empty. Before computing A_k , we adjust D by performing the following procedure.

PROCEDURE ADJUST(D, k)

begin

1. **while** $|D| \geq 2$ and $(f(Tail.last, Tail) < f(Tail, G_k))$ **do** /* Remove paths according to Property 1 */
2. delete *Tail* from D
3. Add G_k to the tail of D
4. **while** $|D| \geq 2$ and $(f(Head, Head.next) > UN[k])$ **do** /* Remove paths according to Property 2 */
5. delete *Head* from D
6. **return**(D)

end.

In our algorithm, $d-1$ calls to ADJUST(D, k) are performed, for $k = 1, 2, \dots, d-1$, respectively. Each G_k is added to D only once, while line 3 of the procedure call ADJUST(D, k) is performed. The addition of each G_k is at the tail of D . Thus, at any time all the paths G_i in D are in an increasing order of i , which guarantees the validity of the deletion in lines 1-2 and 4-5. Since each G_k is added to D only once, there are $d-1$ insertion operations involved in the $d-1$ calls. The total number of deletion operations on D involved in the $d-1$ calls is not larger than the number of insertion operations. Therefore, in total, there are $O(d)$ insertion and deletion operations involved in the $d-1$ calls. Each insertion and deletion on a deque takes

$O(1)$ time. Thus, the $d-1$ calls to $\text{ADJUST}(D, k)$, $k = 1, 2, \dots, d-1$, take $O(d)$ time in total.

After calling $\text{ADJUST}(D, k)$, we compute A_k as the G_i in D minimizing $D(Q_k, G_i)$. The following lemma is very helpful to the computation.

LEMMA 15. *Let $(G_{i_1}, G_{i_2}, \dots, G_{i_r})$ be the path sequence stored in D after calling $\text{ADJUST}(D, k)$, $1 \leq k < d$. If $|D| = 1$ we have $D = (G_k)$; otherwise we have $UN[k] \geq f(G_{i_1}, G_{i_2}) \geq f(G_{i_2}, G_{i_3}) \geq \dots \geq f(G_{i_{r-1}}, G_{i_r})$.*

Proof. We prove this lemma by induction on k . Clearly, $D = (G_1)$ after calling $\text{ADJUST}(D, 1)$. This establishes the lemma for $k = 1$.

Suppose, by induction, that the lemma is true for all values less than k . We show that the lemma holds for k as well. Let $(G_{j_1}, G_{j_2}, \dots, G_{j_s})$ be the path sequence stored in D after calling $\text{ADJUST}(D, k-1)$. We consider two cases: $s = 1$ and $s > 1$.

Case 1. $s = 1$. By the induction hypothesis, $D = (G_{k-1})$ after calling $\text{ADJUST}(D, k-1)$. After performing lines 1–3 of $\text{ADJUST}(D, k)$, we have $D = (G_{k-1}, G_k)$. In lines 4–5, if $UN[k] \geq f(G_{k-1}, G_k)$, no deletion is performed on D ; otherwise G_{k-1} is deleted from D . Clearly, in either case, the lemma holds.

Case 2. $s > 1$. By the induction hypothesis, $UN[k-1] \geq f(G_{j_1}, G_{j_2}) \geq f(G_{j_2}, G_{j_3}) \geq \dots \geq f(G_{j_{s-1}}, G_{j_s})$. Suppose that t , $0 \leq t < s$, paths are deleted from D in lines 1–2 of $\text{ADJUST}(D, k)$. If $t = s-1$, with a proof similar to Case 1, we can show that the lemma holds. Assume that $t < s-1$. According to the loop-repetition condition in line 1, we have $f(G_{j_{s-t-1}}, G_{j_{s-t}}) \geq f(G_{j_{s-t}}, G_k)$. Thus, after line 3, we have $D = (G_{j_1}, G_{j_2}, \dots, G_{j_{s-t}}, G_k)$ and $UN[k-1] \geq f(G_{j_1}, G_{j_2}) \geq f(G_{j_2}, G_{j_3}) \geq \dots \geq f(G_{j_{s-t-1}}, G_{j_{s-t}}) \geq f(G_{j_{s-t}}, G_k)$. Suppose that u , $0 \leq u \leq s-t$, paths are deleted from D in lines 4–5. We have either $u = s-t$ or $UN[k] \geq f(G_{j_{u+1}}, G_{j_{u+2}})$. That is, we have either $D = (G_k)$, or $D = (G_{j_{u+1}}, G_{j_{u+2}}, \dots, G_{j_{s-t}}, G_k)$ and $UN[k] \geq f(G_{j_{u+1}}, G_{j_{u+2}}) \geq f(G_{j_{u+2}}, G_{j_{u+3}}) \geq \dots \geq f(G_{j_{s-t-1}}, G_{j_{s-t}}) \geq f(G_{j_{s-t}}, G_k)$. In either case, the lemma holds. \square

After calling to $\text{ADJUST}(D, k)$, we compute A_k as the path G_i in D minimizing $D(Q_k, G_i)$. In the following, we show that the computation takes $O(1)$ time. Let $(G_{i_1}, G_{i_2}, \dots, G_{i_r})$ be the path sequence stored in D after calling to $\text{ADJUST}(D, k)$. If $|D| = 1$, trivially, we compute A_k as Head in $O(1)$ time. Assume $|D| > 1$. By Lemma 15, we have $UN[k] \geq f(G_{i_1}, G_{i_2}) \geq f(G_{i_2}, G_{i_3}) \geq \dots \geq f(G_{i_{r-1}}, G_{i_r})$. Combining this and Lemma 14, we conclude that $D(Q_k, G_{i_1}) \leq D(Q_k, G_{i_2}) \leq D(Q_k, G_{i_3}) \leq \dots \leq D(Q_k, G_{i_r})$. Thus, in case $|D| > 1$, we can also compute A_k as Head in $O(1)$ time.

We summarize the above discussion in the following algorithm and lemma.

ALGORITHM 3. $\text{ROOTED_1_CORES}(T, C)$

Input: a tree $T = (V, E)$ and a path $C = (c_1, c_2, \dots, c_d)$ in T

Output: a c_1 rooted-1-core of each $T_{c_k}^{c_{k+1}}$, $1 \leq k < d$

begin

1. $(g_1, g_2, \dots, g_{d-1}, L, UN, UD, GD) \leftarrow \text{PREPROCESS}(T, C)$
2. $D \leftarrow \emptyset$
3. **for** each k , $1 \leq k < d$, **do**
4. $D \leftarrow \text{ADJUST}(D, k)$
5. $A_k \leftarrow \text{Head}$ of D .
6. **return** $(A_1, A_2, \dots, A_{d-1})$

end.

LEMMA 16. *Let (c_1, c_2, \dots, c_d) be a path in T . In $O(n)$ time, we can find a c_1 rooted-1-core for every $T_{c_k}^{c_{k+1}}$, $1 \leq k < d$.*

Lemma 16 completes the proof of Theorem 4.

4.2. Finding a 2-core. In this subsection, we show that a 2-core of T can be determined in $O(n)$ time.

Let $C = (c_1, c_2, \dots, c_d)$ be a 1-core of T . Let $S_i, i = 1, 2, \dots, d$, be the d subtrees obtained from T by removing all edges in C such that S_i is the one containing c_i . Becker and Perl [3] gave the following important property for finding a 2-core.

THEOREM 5 (see [3]). *There exists a 2-core $\{A, B\}$ of T satisfying one of the following conditions:*

- (1) $A = C$ and B is a c_i -point core for some subtree $S_i, 2 \leq i \leq d - 1$, or
- (2) (A, B) is a (c_1, c_d) rooted-2-core of T .

On the basis of Theorem 5, the problem of finding a 2-core of a tree can be solved as follows.

ALGORITHM 4. TWO_CORE(T)

Input: a tree $T = (V, E)$

Output: a two-core of T

begin

1. $C = (c_1, c_2, \dots, c_d) \leftarrow$ a 1-core of T
2. **for** each $i, 2 \leq i \leq d - 1$, **do**
3. $R_i \leftarrow$ POINT_CORE(S_i, c_i)
4. $R \leftarrow$ the R_z with $D(C \cup R_z) = \min_{2 \leq i \leq d-1} D(C \cup R_i)$
5. $(X, Y) \leftarrow$ ROOTED_2_CORE(T, c_1, c_d)
6. **if** $D(C \cup R) \leq D(X \cup Y)$
7. **then return** $(\{C, R\})$
8. **else return** $(\{X, Y\})$

end.

The running time of Algorithm 4 is analyzed as follows. By Lemma 2, line 1 takes $O(n)$ time. By Theorem 3, the for-loop in lines 2–3 take $O(\sum_{2 \leq i \leq d-1} |V(S_i)|) = O(n)$ time. Lines 4 and 5, respectively, take $O(d)$ and $O(n)$ time. Lines 6–8 take $O(1)$ time. Therefore, the worst-case time complexity of Algorithm 4 is $O(n)$.

THEOREM 6. *A 2-core of a weighted tree can be found in $O(n)$ time.*

5. A parallel algorithm for computing a 2-core. In this section, we propose a parallel algorithm for computing a 2-core of T on the EREW PRAM. Our parallel algorithm is described as the *work-time presentation framework* proposed by Jájá [11]. In this framework, the performance of a parallel algorithm is measured in terms of two parameters: *parallel running time*, which is the number of time units required to execute the algorithm, and *work*, which is the total number of operations used by the algorithm. The main advantage of the framework is that we do not need to deal with processors. Given a parallel algorithm running in time $R(n)$ and using a total of $W(n)$ operations, this algorithm can be simulated on a k -processor PRAM in $O(W(n)/k + R(n))$ time [11]. Our parallel algorithm for the 2-core problem runs in $O(\log^2 n)$ time using $O(n \log n)$ work on the EREW PRAM. On the CRCW PRAM, the running time can be further improved to $O(\log \log n \log n)$.

Tree contraction [1], list ranking [4], and the Euler-tour technique [25] are well-known techniques for computing tree functions in parallel. They are the major techniques used in this section. Readers not familiar with them may refer to [11] for a clear description. We assume that the data structure representing T is adjacent lists to which the above techniques can be applied efficiently.

A *bottom-up computation tree* is a 4-tuple (T, D, M, F) , where T is a rooted tree whose leaves are labeled with values in D , whose internal nodes are labeled with

operators in M , and whose edges are labeled with functions in F . The value of an internal node v is defined as $val(v) = \oplus(f_1(val(u_1)), f_2(val(u_2)), \dots, f_s(val(u_s)))$, where f_1, f_2, \dots, f_s are the functions associated with the edges connecting v and its children u_1, u_2, \dots, u_s , and \oplus is the operator label of v . A *top-down computation tree* is a 3-tuple (T, D, F) , where T is a rooted tree whose root is labeled with a value in D and whose edges are labeled with functions in F . The value of a node v in this computation tree is $val(v) = f(val(u))$, where u is the parent of v and f is the function associated with the edge (u, v) . Given a bottom-up or top-down computation tree, the *computation tree evaluation problem* is to compute the values of all nodes in the tree. Abrahamson et al. [1] showed that tree contraction is applicable to the computation tree evaluation problem for both bottom-up and top-down computation trees. They gave sufficient conditions for which the computation tree evaluation problem can be solved in $O(\log n)$ time using $O(n)$ work on the EREW PRAM. Please refer to [1] for the details.

The computation of n_u^v, s_u^v, l_u^v , and m_u^v for all $(u, v) \in E$ can be done by giving reduction to the evaluation of computation trees and then applying tree contraction. For example, consider the computation of m_u^v . Assume that all n_u^v are computed and T is rooted at an arbitrary vertex. For each $v \in V$, let $p(v)$ be the parent of v . Computing $m_v^{p(v)}$ for all $v \in V$ can be done by defining a bottom-up computation tree on T according to the following equation:

$$m_v^{p(v)} = d(p(v), v) \times n_v^{p(v)} + \max_{u \text{ is a child of } v} m_u^v.$$

On the other hand, computing $m_{p(v)}^v$ for all $v \in V$ can be done by defining a top-down computation tree on T according to the following equation:

$$m_{p(v)}^v = d(v, p(v)) \times n_{p(v)}^v + \max \left\{ m_{p(p(v))}^{p(v)}, \max_{u \text{ is a child of } p(v), u \neq v} m_u^v \right\}.$$

Note that all the values of $m_v^{p(v)}$ should be computed first such that except $m_{p(p(v))}^{p(v)}$, all quantities at the right-hand side of the above equation are constant. Then, by applying tree contraction twice, once for each computation tree, all m_u^v are computed in $O(\log n)$ time using $O(n)$ work. We have the following lemma.

LEMMA 17. *The values of n_u^v, s_u^v, l_u^v , and m_u^v of every $(u, v) \in E$ can be computed in $O(\log n)$ time using $O(n)$ work.*

We start to present the parallel algorithm. First, we show that Algorithm 1, which was proposed in section 3 for computing an r -point core, can be implemented in $O(\log n)$ time using $O(n)$ work. Using the Euler-tour technique, line 1 orients T into a rooted tree in $O(\log n)$ time using $O(n)$ work [11]. By Lemma 17, line 2 requires the same time and work. Clearly, the values of all $d(r, v)$ can be easily computed by defining a top-down computation tree. For each $v \in V$, we can compute $D(v)$ as $\sum_{u \in N(v)} s_u^v$ by applying list ranking to the adjacent list of v . Thus, line 3 can be done in $O(\log n)$ time using $O(n)$ work. Line 4 computes a u rooted-1-core M_u for every T_u^r , which can be done in $O(\log n)$ time using $O(n)$ work [18]. The implementation of line 5 is slightly complicated. Consider a fixed M_u . Let M_u be $(v_1 = u, v_2, \dots, v_s)$. By using list ranking, two arrays A and B are constructed, where each $A[i]$ stores the vertex v_i , $1 \leq i \leq s$, $B[0]$ stores the edge (r, v_1) , and each $B[i]$ stores the edge (v_i, v_{i+1}) , $1 \leq i < s$. Each $A[i]$ has a key $d(r, v_i)/2$ and each $B[i]$ has a key $d(r, v_i)$. Then we arrange all vertices in A and edges in B into an array C by the increasing order of the keys, which is done by performing an optimal parallel merging algorithm

[11]. Then, for each vertex $B[i]$ in C , a bisector of $P(r, B[i])$ can be computed by finding the nearest edge $A[j]$ to its left in C . The finding is similar to the computation of prefix maximums, which can be done optimally in logarithmic time based upon the balanced binary tree method [11]. Therefore, line 5 requires $O(\log n)$ time using $O(n)$ work. To avoid concurrent accesses in line 9, during the finding of bisectors, each edge (u, v) is associated with the values of $n_u^v, n_v^u, s_u^v, s_v^u, d(r, u)$, and $d(r, v)$ such that while a vertex v obtains its bisector (x_v, y_v) it also obtains copies of $s_{y_v}^{x_v}, n_{y_v}^{x_v}, s_{y_v}^{x_v}, n_{y_v}^{x_v}, d(r, x_v)$, and $d(r, y_v)$. The for-loop in lines 6–10 and the computation in lines 11 and 12 can be easily implemented in $O(\log n)$ time using $O(n)$ work. Therefore, we have the following lemma.

LEMMA 18. *Finding an r -point core of a tree can be done in $O(\log n)$ time using $O(n)$ work on the EREW PRAM.*

Next, let us consider Algorithm 2, which we proposed in subsection 4.1 for finding a (a, b) rooted-2-core of T . In the following, we show that it can be implemented in $O(\log^2 n)$ time using $O(n \log n)$ work. Let $C, Q_i, U_i, S_i, g_i, G_i$, and $A_i, 1 \leq i < d$, be defined as in subsection 4.1. According to Algorithm 2, to show that it can be implemented in $O(\log^2 n)$ time using $O(n \log n)$, we need only to show that computing all $A_k, k = 1, 2, \dots, d-1$, requires the same time and work. Unfortunately, the algorithm we had proposed in subsection 4.1 for the computation is purely sequential. In the following, we present another algorithm for the computation. The presented algorithm takes $O(n \log n)$ sequential time and thus is less efficient. However, it is based upon the divide-and-conquer strategy and thus is of high parallelism.

For each k , let i_k be the smallest integer satisfying $1 \leq i_k \leq k$ and $D(Q_k, G_{i_k}) = \min_{1 \leq i \leq k} D(Q_k, G_i), 1 \leq k < d$. According to Lemma 12, we can compute A_1, A_2, \dots, A_{d-1} , respectively, as $G_{i_1}, G_{i_2}, \dots, G_{i_{d-1}}$. Therefore, the computation of A_1, A_2, \dots, A_{d-1} can be done by determining the values of i_1, i_2, \dots, i_{d-1} . The following lemma is useful for the determination.

LEMMA 19. $i_1 \leq i_2 \leq \dots \leq i_{d-1}$.

Proof. We prove this lemma by showing that $i_k \leq i_{k+1}$ for any fixed $k, 1 \leq k < d-1$. By contradiction, suppose that $i_k > i_{k+1}$. Since $D(Q_{k+1}, G_{i_{k+1}}) = \min_{1 \leq i \leq k+1} \{D(Q_{k+1}, G_i)\}$, we have $D(Q_{k+1}, G_{i_{k+1}}) \leq D(Q_{k+1}, G_{i_k})$, from which we have $f(G_{i_{k+1}}, G_{i_k}) \leq |V(U_{k+1})|$ by Lemma 14. Since $|V(U_{k+1})| < |V(U_k)|$, we have $f(G_{i_{k+1}}, G_{i_k}) < |V(U_k)|$. Therefore, by Lemma 14, $D(Q_k, G_{i_{k+1}}) < D(Q_k, G_{i_k})$, which contradicts the definition of i_k . Thus, $i_k \leq i_{k+1}$ and the lemma holds. \square

Given four integers l, m, x, y such that $1 \leq l < m < d$ and $x \leq i_l \leq i_m \leq y$, we can compute the values of i_l, i_{l+1}, \dots, i_m by the following procedure, which is designed based upon the divide-and-conquer strategy according to Lemma 19.

PROCEDURE. COMPUTE_INDICES(l, m, x, y)

begin

1. $z \leftarrow \lfloor (l+m)/2 \rfloor$

2. $i_z \leftarrow$ the smallest integer satisfying $x \leq i_z \leq y, i_z \leq z$, and $D(Q_z, G_{i_z})$ is minimized

3. **if** $l < z$ **then** COMPUTE_INDICES($l, z-1, x, i_z$)

4. **if** $z < m$ **then** COMPUTE_INDICES($z+1, m, i_z, y$)

end.

Recall that using arrays L, UN, UD , and GD , the value of $D(Q_k, G_i)$ can be computed in $O(1)$ sequential time for any two given integers k and $i, 1 \leq i \leq k < d$. Therefore, assuming that the arrays are computed, it is easy to see that the above procedure takes $O((y-x) \times \log(m-l))$ sequential time. Line 2 determines the

minimum among $O(y-x)$ values, which can be done in $O(\log(y-x))$ time using $O(y-x)$ work. The two recursive calls, respectively, in lines 3 and 4 can be performed in parallel. Therefore, with some efforts, we can show that the above procedure can be implemented in $O(\log(y-x) \times \log(m-l))$ time using $O((y-x) \times \log(m-l))$ work on the EREW PRAM. Note that a careful implementation of line 2 is required for avoiding concurrent accesses to the arrays L , UN , UD , and GD . We omit the details here.

Our new algorithm for computing $A_k, k = 1, 2, \dots, d-1$, is as follows.

ALGORITHM 5. D&C_ROOTED_1CORES(T, C)

Input: a tree $T = (V, E)$ and a path $C = (c_1, c_2, \dots, c_d)$

Output: a c_1 rooted-1-core of each $T_{c_k}^{c_{k+1}}, 1 \leq k < d$

begin

1. $(g_1, g_2, \dots, g_{d-1}, L, UN, UD, GD) \leftarrow \text{PREPROCESS}(T, C)$

2. COMPUTE_INDICES $(1, d-1, 1, d-1)$

3. **return** $(G_{i_1}, G_{i_2}, \dots, G_{i_{d-1}})$

end.

Clearly, Algorithm 5 takes $O(n \log n)$ sequential time. Its performance on the EREW PRAM is analyzed as follows. PREPROCESS(T, C) can be easily implemented in $O(\log n)$ time using $O(n)$ work. Line 2 requires $O(\log^2 d)$ time using $O(d \log d)$ work. Therefore, Algorithm 5 runs in $O(\log^2 n)$ time using $O(n \log n)$ work. Applying this result to Algorithm 2, we obtain the following lemma.

LEMMA 20. *A (a, b) rooted-2-core of a tree can be found in $O(\log^2 n)$ time using $O(n \log n)$ work on the EREW PRAM.*

A 1-core of a tree can be found in $O(\log n)$ time using $O(n)$ work [18]. By applying this result and Lemmas 18 and 20 to Algorithm 4, which we proposed for computing a 2-core, we obtain the following theorem.

THEOREM 7. *A 2-core of a weighted tree can be computed in $O(\log^2 n)$ time using $O(n \log n)$ work on the EREW PRAM.*

On the CRCW PRAM model, the minimum of a set of n elements can be found in $O(\log \log n)$ time using $O(n)$ work [11]. In case the n elements are integers bounded by a polynomial in n , the parallel running time can be further reduced to $O(1)$ without increasing the work [11]. Applying these results to line 2 of COMPUTE_INDICES, we can obtain the following theorems.

THEOREM 8. *A 2-core of a weighted tree can be computed in $O(\log \log n \log n)$ time using $O(n \log n)$ work on the CRCW PRAM.*

THEOREM 9. *Given an unweighted tree T , a 2-core of T can be computed in $O(\log n)$ time using $O(n \log n)$ work on the CRCW PRAM.*

6. Concluding remarks. The definition of the p -core problem restricts the p paths selected as a p -core to be mutually disjoint. In case the restriction is removed the p -core problem becomes an easy one. Define an *intersection p -core* of T as a set of p paths $\{I_1, I_2, \dots, I_p\}$ in T minimizing $D(I_1 \cup I_2 \cup \dots \cup I_p)$. Becker and Perl [3] had studied the case $p = 2$ of the intersection p -core problem and gave an $O(n^2)$ time algorithm. In the following, we show that the intersection p -core problem can be solved in linear time, even when p is a variable.

Let $\{I_1, I_2, \dots, I_p\}$ be an intersection p -core of T . Let S be the subtree of T induced by the vertices in $V(I_1) \cup V(I_2) \cup \dots \cup V(I_p)$. Clearly, S has at most $2p$ leaves and $S \supseteq (I_1 \cup I_2 \cup \dots \cup I_p)$. Since S has at most $2p$ leaves, it is easy to see that we can find a set of p paths Y_1, Y_2, \dots, Y_p such that $S = Y_1 \cup Y_2 \cup \dots \cup Y_p$. Note that $Y_i, i = 1, 2, \dots, p$, are not necessary to be mutually disjoint. Since $S \supseteq (I_1 \cup I_2 \cup \dots \cup I_p)$, we have $D(Y_1 \cup Y_2 \cup \dots \cup Y_p) = D(S) \leq D(I_1 \cup I_2 \cup \dots \cup I_p)$. Since $\{I_1, I_2, \dots, I_p\}$ is

an intersection p -core, we have $D(Y_1 \cup Y_2 \cup \dots \cup Y_p) = D(I_1 \cup I_2 \cup \dots \cup I_p)$ and thus $\{Y_1, Y_2, \dots, Y_p\}$ is also an intersection p -core. From the above discussion, we conclude that the problem of finding an intersection p -core of T is equivalent to the problem of finding a subtree S of T such that S has at most $2p$ leaves and S minimizes $D(S)$ over all subtrees of T having at most $2p$ leaves. Shioura and Uno [21] had solved the latter problem in $O(n)$ time. Furthermore, Wang [27] solved the problem in $O(\log n \log^* n)$ time using $O(n)$ work on the EREW PRAM. Thus, the intersection p -core problem can be solved in $O(\log n \log^* n)$ time using $O(n)$ work on the EREW PRAM.

Finally, we conclude this paper by giving directions for further studies. One direction is to solve the 2-core problem in polylogarithmic time using $O(n)$ work on the PRAM. Another direction is to design an $o(n^2)$ time p -core algorithm for constant $p > 2$. Hakimi, Schmeichel, and Labbe [9] showed that after the addition of length-constraint, the general p -core problem becomes NP-hard. One direction for further studies is to find an efficient approximation algorithm for the constrained problem. Hakimi, Schmeichel, and Labbe's proof is under the assumption that T is weighted. Therefore, to study the complexity of the constrained problem on an unweighted tree is also a possible direction.

Acknowledgments. The author expresses his gratitude to the anonymous referees for their valuable suggestions and to Prof. Arie Tamir for pointing out that the general p -core problem had been solved in $O(pn^2)$ time by Novik in [17].

REFERENCES

- [1] K. ABRAHAMSON, N. DADOUN, D. G. KIRKPATRICK, AND T. PRZYTYCKA, *A simple parallel tree contraction algorithm*, J. Algorithms, 10 (1989), pp. 287–302.
- [2] S. ALSTRUP, P. W. LAURIDSEN, P. SOMMERLUND, AND M. THROUP, *Finding cores of limited length*, in Proceedings of the 5th International Workshop on Algorithms and Data Structures, Lecture Notes in Comput. Sci. 1272, Springer-Verlag, Berlin, 1997, pp. 45–54.
- [3] R. I. BECKER AND Y. PERL, *Finding the two-core of a tree*, Discrete Appl. Math., 11 (1985), pp. 103–113.
- [4] R. COLE AND U. VISHKIN, *Approximate parallel scheduling, Part I: The basic technique with applications to optimal parallel list ranking in logarithmic time*, SIAM J. Comput., 17 (1988), pp. 128–142.
- [5] G. N. FREDERICKSON AND D. B. JOHNSON, *Finding k th paths and p -centers by generating and searching good data structures*, J. Algorithms, 4 (1983), pp. 61–80.
- [6] B. GAVISH AND S. SRIDHAR, *Computing the 2-median on tree networks in $O(n \lg n)$ time*, Networks, 26 (1995), pp. 305–317.
- [7] A. J. GOLDMAN, *Optimal center location in simple networks*, Transportation Sci., 5 (1971), pp. 212–221.
- [8] S. L. HAKIMI, *Optimal distribution of switching centers in communication networks and some related graph theoretical problems*, Oper. Res., 13 (1965), pp. 462–475.
- [9] S. L. HAKIMI, E. F. SCHMEICHEL, AND M. LABBE, *On locating path- or tree-shaped facilities on networks*, Networks, 23 (1993), pp. 543–555.
- [10] G. Y. HANDLER AND P. MIRCHANDANI, *Location on Networks*, MIT Press, Cambridge, MA, 1979.
- [11] J. JÁJÁ, *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA, 1992.
- [12] D. E. KNUTH, *The Art of Computer Programming*, Vol. 1, Addison-Wesley, Reading, MA, 1968.
- [13] N. MEGIDDO, A. TAMIR, E. ZEMEL, AND R. CHANDRASEKARAN, *An $O(n \log^2 n)$ algorithm for the k th longest path in a tree with applications to location problems*, SIAM J. Comput., 10 (1981), pp. 328–337.
- [14] E. MINIEKA, *The optimal location of a path or tree in a tree network*, Networks, 15 (1985), pp. 309–321.
- [15] E. MINIEKA AND N. H. PATEL, *On finding the core of a tree with a specified length*, J. Algorithms, 4 (1983), pp. 345–352.
- [16] C. A. MORGAN AND P. L. SLATER, *A linear time algorithm for a core of a tree*, J. Algorithms,

- 1 (1980), pp. 247–258.
- [17] A. NOVIK, *Improved Algorithms for Locating Tree or Path Shaped Facilities on a Tree Network*, M.S. Thesis, School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel, 1996.
 - [18] S. PENG AND W.-T. LO, *A simple optimal parallel algorithm for a core of a tree*, J. Parallel Distrib. Comput., 20 (1994), pp. 388–392.
 - [19] S. PENG, A. B. STEPHEN, AND Y. YESHA, *Algorithms for a core and k -tree core of a tree*, J. Algorithms, 15 (1993), pp. 143–159.
 - [20] S. PENG AND W. LO, *Efficient algorithms for finding a core of a tree with specified length*, J. Algorithms, 20 (1996), pp. 445–458.
 - [21] A. SHIOURA AND T. UNO, *A linear time algorithm for finding a k -tree core*, J. Algorithms, 23 (1997), pp. 281–290.
 - [22] P. J. SLATER, *On locating a facility to service areas within a network*, Oper. Res., 29 (1981), pp. 523–531.
 - [23] P. J. SLATER, *Locating central paths in a network*, Transportation Sci., 16 (1982), pp. 1–18.
 - [24] B. C. TANSEL, R. L. FRANCIS, AND T. J. LOWE, *Location on networks: A survey*, Management Sci., 29 (1983), pp. 482–511.
 - [25] R. E. TARJAN AND U. VISHKIN, *An efficient parallel biconnectivity algorithm*, SIAM J. Comput., 14 (1985), pp. 862–874.
 - [26] A. TAMIR, *An $O(pn^2)$ time algorithm for the p -median and related problems on tree graphs*, Oper. Res. Lett., 19 (1996), pp. 59–64.
 - [27] B.-F. WANG, *Finding a k -tree core and a k -tree center of a tree network in parallel*, IEEE Trans. Parallel Distrib. Systems, 9 (1999), pp. 186–191.
 - [28] B.-F. WANG, *Efficient parallel algorithms for optimally locating a path and a tree of a specified length in a weighted tree network*, J. Algorithms, 34 (2000), pp. 90–108.
 - [29] B.-F. WANG, S.-C. KU, AND K.-H. SHI, *Cost-optimal parallel algorithms for the tree bisector and related problems*, IEEE Trans. Parallel Distrib. Systems, 12 (2001), pp. 888–898.

TESTING k -COLORABILITY*

NOGA ALON[†] AND MICHAEL KRIVELEVICH[†]

Abstract. Let G be a graph on n vertices and suppose that at least ϵn^2 edges have to be deleted from it to make it k -colorable. It is shown that in this case most induced subgraphs of G on $ck \ln k / \epsilon^2$ vertices are not k -colorable, where $c > 0$ is an absolute constant. If G is as above for $k = 2$, then most induced subgraphs on $\frac{(\ln(1/\epsilon))^b}{\epsilon}$ are nonbipartite, for some absolute positive constant b , and this is tight up to the polylogarithmic factor. Both results are motivated by the study of testing algorithms for k -colorability, first considered by Goldreich, Goldwasser, and Ron in [*J. ACM*, 45 (1998), pp. 653–750], and improve the results in that paper.

Key words. graph coloring, property testing

AMS subject classifications. 05C15, 68R10

PII. S0895480199358655

1. Introduction. Suppose that for a fixed integer k and a small $\epsilon > 0$, a graph $G = (V, E)$ on n vertices is such that at least ϵn^2 edges should be deleted to make G k -colorable. Clearly G contains many non- k -colorable subgraphs. Some of them are probably quite small in order. What is then the smallest non- k -colorable subgraph of G ? How many small non- k -colorable subgraphs are there?

In order to address the above questions quantitatively, we introduce a suitable notation. First, we call a graph G on n vertices ϵ -robustly non- k -colorable or alternatively ϵ -far from being k -colorable if after deleting any subset of less than ϵn^2 edges of G the remaining graph is still not k -colorable. Of course, it follows that G itself is not k -colorable. Define

$$f_k(G) = \min\{|V_0| : V_0 \subseteq V(G), G[V_0] \text{ is non-}k\text{-colorable}\},$$

where $G[V_0]$ denotes the subgraph of G induced by V_0 . If $\chi(G) \leq k$, we set $f_k(G) = \infty$. For an integer n and $0 < \epsilon < 1/(2k)$ let

$$f_k(n, \epsilon) = \max\{f_k(G) : G \text{ is an } \epsilon\text{-robustly non-}k\text{-colorable graph on } n \text{ vertices}\}.$$

(Note that the assumption $\epsilon < 1/(2k)$ can be made without loss of generality as every graph on n vertices is at most $n^2/(2k)$ edges far from being k -colorable). Similarly, let

$$g_k(G) = \min\{t : \text{if } R \subseteq V(G) \text{ is chosen uniformly at random from all subsets of } V \text{ of size } t, \text{ then } \Pr[\chi(G[R]) > k] \geq 1/2\}.$$

*Received by the editors July 6, 1999; accepted for publication (in revised form) January 28, 2002; published electronically March 20, 2002.

<http://www.siam.org/journals/sidma/15-2/35865.html>

[†]Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel (noga@math.tau.ac.il, krivelev@math.tau.ac.il). The first author's research was supported in part by a USA-Israeli BSF grant, by the Israel Science Foundation, and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University. The second author's research was partially performed while the author was with DIMACS Center, Rutgers University, Piscataway, NJ, and was supported by a DIMACS Postdoctoral Fellowship, by a USA-Israeli BSF Grant, and by a Bergmann Memorial Grant.

Again, $g_k(G) = \infty$ if $\chi(G) \leq k$. Let

$$g_k(n, \epsilon) = \max\{g_k(G) : G \text{ is an } \epsilon\text{-robustly non-}k\text{-colorable graph on } n \text{ vertices}\} .$$

Obviously, $f_k(G) \leq g_k(G)$ for any graph G , thus implying $f_k(n, \epsilon) \leq g_k(n, \epsilon)$.

A few comments on the above definitions are in order. The function $f_k(n, \epsilon)$ represents a very natural extremal graph theory problem, seeking to link the size of a smallest non- k -colorable subgraph of a non- k -colorable graph with its distance from the set of k -colorable graphs. For example, for $k = 2$ one can say that if the odd girth (i.e., the minimal length of an odd cycle) of a graph G on n vertices is more than $f_2(n, \epsilon)$ for some $\epsilon > 0$, then G can be made bipartite by deleting less than ϵn^2 edges. The function $g_k(n, \epsilon)$ says that if G is ϵ -robustly non- k -colorable, then it contains not only one but very many non- k -colorable subgraphs on $g_k(n, \epsilon)$ vertices. The somewhat artificial looking definition of $g_k(n, \epsilon)$ actually has a very natural algorithmic background in terms of graph property testing, as considered by Goldreich, Goldwasser, and Ron in [3]. Applied to the particular problem of testing k -colorability, their approach reads as follows. Suppose our aim is to design an algorithm, which for a given (large enough) integer n and a (small enough) parameter $\epsilon > 0$, distinguishes with high probability between an input graph on n vertices, which is k -colorable, and that in which at least ϵn^2 edges should be deleted to create a k -colorable graph. The algorithm can query whether or not a specific pair of vertices of the input graph is connected by an edge. In general, it is NP-complete to check k -colorability for any $k \geq 3$. However, given the assumption that the input is either k -colorable or very far from it, one may hope to devise very efficient randomized algorithms. We refer the reader to [3] for a general discussion of graph property testing.

Returning to the definition of the function $g_k(n, \epsilon)$, one can propose the following very simple algorithm for testing k -colorability. Given an input graph $G = (V, E)$, choose uniformly at random $g_k(n, \epsilon)$ vertices of G and denote the chosen set by R . Now, check whether the induced subgraph $G[R]$ is k -colorable. If it is, output “ G is k -colorable”; otherwise output “ G is not- k -colorable.” Note that if G is k -colorable, then every subgraph of it is k -colorable as well. Thus, in this case we *always* output a correct answer. On the other hand, if G is ϵ -far from being k -colorable, it follows from the definition of $g_k(n, \epsilon)$ that a sample of size $g_k(n, \epsilon)$ induces a non- k -colorable graph with probability at least $1/2$. Therefore, in this case we output a correct answer with probability at least $1/2$. Having in mind the above discussion, sometimes we will refer to bounding the function $g_k(n, \epsilon)$ as *testing k -colorability*.

The problem of estimating $f_k(n, \epsilon)$ and $g_k(n, \epsilon)$ will be treated in this paper as an *asymptotic* one. This means that whenever needed we will assume that the number of vertices n is large enough, and that the robustness parameter ϵ is small enough, but fixed as n is growing.

It is important to observe that the values of the functions defined above are of interest only for graphs with a *quadratic* number of edges. Indeed, if G has n vertices and is ϵ -far from being k -colorable, it contains at least ϵn^2 edges. This observation, together with the asymptotic nature of the problem, prompts the use of graph theoretic methods designed for dense graphs, most notably the well-known regularity lemma of Szemerédi [6].

Let us now survey the previous research on these problems. Somewhat surprisingly it turned out that the above defined function $g_k(n, \epsilon)$ can be bounded from above by a function of ϵ only. This has been proven by Bollobás et al. [2] for the case

$k = 2$ and by Rödl and Duke [5] for every $k \geq 3$. Both papers rely on the regularity lemma. As is the case with most applications of the regularity lemma, the resulting bounds are extremely fast growing functions of $1/\epsilon$ (towers of height polynomial in $1/\epsilon$), thus making the results hardly applicable from the practical point of view. Note that both papers [2] and [5] formulate their results in a somewhat different language and do not define the function $g_k(n, \epsilon)$ explicitly.

For $k = 2$, Komlós showed in [4] that $f_2(n, \epsilon) = O(1/\epsilon^{1/2})$. This result is easily seen to be tight by considering a blow-up of an odd cycle of length about $1/\epsilon^{1/2}$. (A graph G on n vertices is a *blow-up* of a graph H on m vertices with vertex set $V(H) = \{v_1, \dots, v_m\}$ if the vertex set of G can be partitioned into m disjoint sets V_1, \dots, V_m , each of size $|V_i| = n/m$, so that V_i and V_j are joined completely if $(v_i, v_j) \in E(H)$ and are not joined by any edge otherwise.)

Motivated by testing k -colorability, Goldreich, Goldwasser, and Ron [3] came up with a completely different approach for bounding $g_k(n, \epsilon)$. Using direct combinatorial arguments (and thus avoiding the regularity lemma), they were able to prove that $g_2(n, \epsilon) = O(\log(1/\epsilon)/\epsilon^2)$ —a tremendous progress compared to the bound of [2]. Similarly, they proved that for every fixed $k \geq 3$ one has $g_k(n, \epsilon) = O(k^2 \log k/\epsilon^3)$. The difference between the cases $k = 2$ and $k \geq 3$ can be intuitively explained by the fact that for $k = 2$ the family of minimal non-2-colorable graphs coincides with the family of odd cycles and is thus very simple to describe. For every $k \geq 3$ the family of minimal non- k -colorable graphs (usually called $(k + 1)$ -color-critical graphs) is very complicated. Goldreich, Goldwasser, and Ron did not discuss the function $f_k(n, \epsilon)$ and did not provide any separate bounds for it.

The purpose of the present paper is to provide improved bounds for both functions f_k and g_k . We prove the following results.

THEOREM 1.

1. For all $\epsilon \leq 1/9$, $g_2(n, \epsilon) \geq \frac{1}{6\epsilon}$.
2. For every fixed $k \geq 3$ and every small enough $\epsilon > 0$, for infinitely many n one has

$$g_k(n, \epsilon) \geq f_k(n, \epsilon) \geq \frac{1}{110} \left(\frac{1}{330 \ln k} \right)^{\frac{2}{k-2}} \frac{1}{\epsilon}.$$

THEOREM 2.

$$g_2(n, \epsilon) \leq \frac{34 \ln^4 \left(\frac{1}{\epsilon} \right) \ln \ln \left(\frac{1}{\epsilon} \right)}{\epsilon}.$$

THEOREM 3. For every fixed $k \geq 3$,

$$g_k(n, \epsilon) \leq \frac{36k \ln k}{\epsilon^2}.$$

These results improve upon the above mentioned bounds of Goldreich, Goldwasser, and Ron [3]. Still, for every $k \geq 3$, the gap between the lower and the upper bounds, given by Theorems 1 and 3, respectively, remains quite substantial.

The rest of the paper is organized as follows. In section 2 we discuss lower bounds for the functions f_k, g_k and prove Theorem 1. In section 3 we prove Theorem 2. Section 4 is devoted to proving Theorem 3. Section 5 contains some concluding remarks and open problems.

During the course of the proof we make no serious attempts to optimize the constants involved. Also, we omit routinely floor and ceiling signs to simplify the presentation. Given a graph $G = (V, E)$ and a vertex $v \in V$, we denote by $N(v)$ the set of all neighbors of v in G . The degree of v in G is denoted by $d(v)$. For a vertex $v \in V$ and a subset $U \subset V$, we denote by $d(v, U)$ the number of neighbors of v in U . The number of edges of G spanned by U , i.e., having both endpoints in U , is denoted by $e(U)$. A vertex $v \in V(G)$ is *dominated* by a subset $S \subseteq V(G)$ if v has a neighbor inside S in G . Recall that, whenever needed, the number of vertices n is assumed to be large enough, while $\epsilon > 0$ is small enough.

2. Lower bounds. In this section we prove lower bounds for the functions f_k, g_k . For many graph testing problems, a lower bound of order $1/\epsilon$ is very natural and can be proven quite easily. The property of k -colorability is not an exception, and the bound $g_k(n, \epsilon) \geq c(k)/\epsilon$ can be obtained by considering a complete $(k + 1)$ -partite graph with one part of size $\Theta(\epsilon n)$ and the other k of equal size. This is how we prove Theorem 1, part 1. For every $k \geq 3$ we prove a stronger statement. Namely, we show the existence of an ϵ -robustly non- k -colorable graph on n vertices in which, for a fixed constant $c = c(k)$, not only does a typical subset of size c/ϵ induce a k -colorable graph but *every* subgraph of this order is k -colorable. This is done by considering a random graph with a linear number of edges and then blowing it up to get an ϵ -robustly non- k -colorable graph which is locally k -colorable. This supplies a lower bound for the function $f_k(n, \epsilon)$. It is worth noting here that the case $k = 2$ is different, as it follows from the result of Komlós [4] that $f_2(n, \epsilon) = \Theta(1/\epsilon^{1/2})$.

Proof of Theorem 1, part 1. Given n, ϵ , let G be a complete tripartite graph with parts V_0, V_1, V_2 of sizes $|V_0| = 3\epsilon n, |V_1| = |V_2| = \frac{1-3\epsilon}{2}n$. Notice that each edge of G participates in at most $(1 - 3\epsilon)n/2$ triangles. As the total number of triangles in G is $3\epsilon(1 - 3\epsilon)^2n^3/4$, at least $3\epsilon(1 - 3\epsilon)n^2/2 \geq \epsilon n^2$ edges should be deleted to destroy all the triangles of G . Therefore, G is ϵ -robustly non-2-colorable. In order to estimate $g_2(G)$, note that if $R \subset V(G)$ is such that $R \cap V_0 = \emptyset$, then the subgraph $G[R]$ is bipartite. Thus, in order to have $\chi(G[R]) = 3$, the set R has to hit V_0 . If R is chosen uniformly at random from all subsets of $V(G)$ of size r , then

$$Pr[R \cap V_0 \neq \emptyset] \leq \frac{|V_0| \binom{n-1}{r-1}}{\binom{n}{r}} = \frac{|V_0|r}{n} = 3\epsilon r .$$

Thus, requiring $Pr[\chi(G[R]) = 3] \geq 1/2$ implies $3\epsilon r \geq 1/2$, which in turn gives $r \geq 1/(6\epsilon)$. As G is ϵ -robustly nonbipartite, the statement follows. \square

Proof of Theorem 1, Part 2. Let us define

$$\begin{aligned} c_1 &= c_1(k) = \left(\frac{1}{3}\right)^{\frac{2}{k-2}} \left(\frac{1}{40e \ln k}\right)^{\frac{k}{k-2}} , \\ c_2 &= c_2(k) = 2 \ln k , \\ c_3 &= c_3(k) = 40k \ln k . \end{aligned}$$

The key ingredient of the proof is the following lemma.

LEMMA 2.1. *For every fixed $k \geq 3$ and a sufficiently large integer m , there exists a graph $H = H_{k,m}$ on m vertices having the following properties:*

1. *Every subset of c_1m vertices of H spans a k -colorable graph.*
2. *Every subset $U \subset V(H)$ of size $|U| > m/(2k)$ spans at least $c_2|U|$ edges.*
3. *At least $c_2m/2$ edges need to be deleted from H to create a k -colorable graph.*

Proof. Set $p = p(m) = c_3/m$ and consider the random graph $G(m, p)$. This is a random graph with vertex set $\{1, \dots, m\}$ in which every pair $1 \leq i < j \leq m$ is an edge independently and with probability p . We will prove that almost surely $G(m, p)$ has the desired properties. In this proof the term “almost surely” (or a.s. for short) means that the probability that all desired properties hold tends to 1 as $m \rightarrow \infty$.

In order to prove that the first assertion of the lemma holds a.s. for the random graph $G(m, p)$, note that a non- k -colorable graph contains a subgraph in which all degrees are at least k . Thus, if the first assertion fails, the random graph contains a subset U of size $|U| \leq c_1 m$, spanning at least $(k/2)|U|$ edges. The probability of this event can be bounded from above by the following expression:

$$\sum_{i=k+1}^{c_1 m} \binom{m}{i} \binom{\binom{i}{2}}{\frac{k}{2}i} p^{\frac{k}{2}i} < \sum_{i=k+1}^{c_1 m} \left(\frac{em}{i}\right)^i \left(\frac{ei}{k}\right)^{\frac{ki}{2}} p^{\frac{ki}{2}} = \sum_{i=k+1}^{c_1 m} \left[\frac{em}{i} \left(\frac{eip}{k}\right)^{\frac{k}{2}}\right]^i.$$

Denote the i th summand of the last sum by a_i . Then, if $m^{1/2} \leq i \leq c_1 m$ we have

$$\begin{aligned} a_i &\leq \left[\frac{em}{c_1 m} \left(\frac{ec_1 c_3 m}{km}\right)^{\frac{k}{2}}\right]^i = \left[\frac{e}{c_1} \left(\frac{ec_1 c_3}{k}\right)^{\frac{k}{2}}\right]^i = \left[e \left(\frac{ec_3}{k}\right)^{\frac{k}{2}} c_1^{\frac{k}{2}-1}\right]^i \\ &= \left[e(40e \ln k)^{\frac{k}{2}} \frac{1}{3} \left(\frac{1}{40e \ln k}\right)^{\frac{k}{2}}\right]^i = \left(\frac{e}{3}\right)^i = o(m^{-1}). \end{aligned}$$

If $4 \leq i < m^{1/2}$, we get

$$a_i < \left[em^{1/2} \left(\frac{ec_3}{km^{1/2}}\right)^{\frac{k}{2}}\right]^4 = \left(\frac{e^{\frac{k}{2}+1} (40 \ln k)^{\frac{k}{2}}}{m^{\frac{k}{4}-\frac{1}{2}}}\right)^4 = o(m^{-1/2}).$$

Thus, $\sum_{i=k+1}^{c_1 m} a_i = o(1)$, showing that the first part of the lemma holds with high probability in $G(m, p)$.

For the second part of the lemma, note that for a fixed subset $U \subseteq V(G(m, p))$ of size $|U| = i$, the number of edges spanned by U in $G(m, p)$ is a binomial random variable with parameters $\binom{i}{2}$ and p . Using the well-known Chernoff bounds on the tails of binomial distribution (cf., e.g., [1, Appendix A]), we get $Pr[|E(G[U])| < \binom{i}{2}p - a] < \exp\{-a^2/(2\binom{i}{2}p)\}$. Therefore, the probability of existence of a subset U , violating the assertion of the second part of the lemma, can be bounded from above by

$$\begin{aligned} &\sum_{i>m/2k} \binom{m}{i} \exp\left\{-\frac{((\binom{i}{2})p - c_2 i)^2}{2\binom{i}{2}p}\right\} < \sum_{i>m/2k} \left(\frac{em}{i}\right)^i \exp\left\{-\frac{(\frac{i-1}{2}p - c_2)^2 i}{(i-1)p}\right\} \\ &< \sum_{i>m/2k} (2ek)^i \exp\left\{-\frac{m\left(\frac{c_3}{2} \frac{i-1}{m} - c_2\right)^2}{c_3}\right\}. \end{aligned}$$

Denote the i th summand in the sum above by b_i . Notice that $c_2 = c_3/(20k) \leq (1/5)(i-1)c_3/(2m)$ for $i > m/(2k)$. Hence

$$b_i < (2ek)^i e^{-\frac{m}{c_3} \left(\frac{2c_3(i-1)}{5m}\right)^2} < e^{\ln(2ek)i - \frac{4c_3(i-1)^2}{25m}} < e^{3i \ln k - 3.2(i-1) \ln k} = o(m^{-1}).$$

Finally, we prove the third part of the lemma. Let $V(H) = C_1 \cup \dots \cup C_k$ be a k -partition of the vertex set of H . Then, by part 2 of the lemma,

$$\begin{aligned} \sum_{j:|C_j|>\frac{m}{2k}} |\{(u, v) \in E(H) : u, v \in C_j\}| &\geq \sum_{j:|C_j|>\frac{m}{2k}} c_2 \cdot |C_j| \\ &= c_2 m - \sum_{j:|C_j|\leq\frac{m}{2k}} c_2 \cdot |C_j| \geq \frac{c_2 m}{2}. \end{aligned}$$

We have thus proven that the random graph $G(m, p)$, with p as defined above, has a.s. the desired properties. \square

In order to prove Theorem 1, part 2, we take the output of Lemma 2.1 and blow it up to show the existence of a graph with the desired properties. Set

$$m = \left\lfloor \frac{c_2}{2\epsilon} \right\rfloor = \left\lfloor \frac{\ln k}{\epsilon} \right\rfloor.$$

Assume that $\epsilon > 0$ is such that the conclusion of Lemma 2.1 holds for $m = m(\epsilon)$ as defined above. Let $H = H_{k,m}$ be the graph from Lemma 2.1. Label the vertices of H by the integers $1, \dots, m$. For an integer n divisible by m , define a graph $G = (V, E)$ on n vertices as follows. The vertex set $V(G)$ is a union of m disjoint subsets V_1, \dots, V_m , each of size n/m . For each pair $1 \leq i \neq j \leq m$, vertices $u \in V_i, v \in V_j$ are connected by an edge in G if and only if $(i, j) \in E(H)$.

Let us now state some properties of the obtained graph G . First, G is easily seen to be homomorphic to H . (We say that G_1 is *homomorphic* to G_2 if there exists a mapping $\phi : V(G_1) \rightarrow V(G_2)$ so that for every edge $(u, v) \in E(G_1)$, $(\phi(u), \phi(v)) \in E(G_2)$.) Therefore, every subgraph of G is homomorphic to a subgraph of H . As a homomorphism does not decrease the chromatic number, we derive from Lemma 2.1 that every subgraph of G on at most $c_1 m$ vertices is k -colorable.

Next, we need to estimate the distance from G to the set of k -colorable graphs on n vertices. Let $V = C_1 \cup \dots \cup C_k$ be a k -partition of $V(G)$ with a minimal number of monochromatic edges. Denote the latter by s . Consider a random k -partition of $V(H)$ induced by assigning a color j , $1 \leq j \leq k$, to a vertex i , $1 \leq i \leq m$, with probability $|C_j \cap V_i|/|V_i|$. The expected number of monochromatic edges of H under such a partition is

$$\begin{aligned} \sum_{(i_1, i_2) \in E(H)} \sum_{j=1}^k \frac{|C_j \cap V_{i_1}|}{|V_{i_1}|} \frac{|C_j \cap V_{i_2}|}{|V_{i_2}|} &= \frac{1}{(n/m)^2} \sum_{j=1}^k \sum_{(i_1, i_2) \in E(H)} |C_j \cap V_{i_1}| |C_j \cap V_{i_2}| \\ &= \frac{m^2}{n^2} \sum_{j=1}^k |\{(u, v) \in E(G) : u, v \in C_j\}| = \frac{m^2 s}{n^2}. \end{aligned}$$

As by our assumption on H we have that the distance from H to the family of k -colorable graphs on m vertices is at least $c_2 m/2$, we get $s \geq c_2 n^2/(2m)$.

Recalling now the definitions of m and of the constants c_1, c_2 , we conclude that G has the following properties:

1. G is ϵ -robustly non- k -colorable.
2. Every subgraph of G on at most $c_1 m = \frac{c_1 c_2}{2\epsilon} = \frac{1}{40e} \left(\frac{1}{120e \ln k}\right)^{\frac{2}{k-2}} \frac{1}{\epsilon}$ vertices is k -colorable.

This implies Theorem 1, part 2. \square

3. Testing bipartiteness. In this section we prove Theorem 2. Our proof exploits the basic elegant idea of Goldreich, Goldwasser, and Ron [3]. It is, however, far more involved technically.

Let us first describe briefly the main idea of the argument of Goldreich, Goldwasser, and Ron for testing bipartiteness. Let $G = (V, E)$ be an ϵ -robustly non-bipartite graph on n vertices. We need to show that a random sample R of size $|R| = \tilde{O}(1/\epsilon^2)$ contains a non-2-colorable subgraph (i.e. an odd cycle) with probability at least $1/2$. The set R will be generated in two stages: $R = S \cup T$, where S is a random subset of size $|S| = \tilde{O}(1/\epsilon)$ and T is a random subset of size $|T| = \tilde{O}(1/\epsilon^2)$. First, it is easy to see that with probability at least $3/4$ such S as above will dominate most of the vertices of G of degree at least $\epsilon n/2$. We assume that S indeed has this property. For a partition $S = S^1 \cup S^2$, denote by U^1 the set of vertices of G of degree at least $\epsilon n/2$, dominated by S^1 ; also let U^2 be the remaining vertices of degree at least $\epsilon n/2$, dominated by S . We call any edge $e \in E(G)$ spanned by U^1 or by U^2 a *witness* for the partition $S = S^1 \cup S^2$. If a random set T contains a witness for every partition $S = S^1 \cup S^2$, then the union $S \cup T$ is easily seen to span a nonbipartite subgraph.

Recall that G is ϵ -robustly nonbipartite. Therefore, for every partition $S = S^1 \cup S^2$, dominating most of the vertices of degree at least $\epsilon n/2$, at least one of the sets U^1, U^2 should span at least $\epsilon n^2/4$ edges, each of them being a witness for $S^1 \cup S^2$. If we choose the vertices of T of size $|T| = \tilde{O}(1/\epsilon^2)$ pair after pair, then the probability that T does not contain a witness for a fixed partition $S^1 \cup S^2$ is at most $2^{-\tilde{\Omega}(1/\epsilon)} \ll 2^{-|S|}$. As S has $2^{|S|}$ partitions, by the union bound we obtain that the probability that T does not contain a witness for one of the partitions is much less than $2^{|S|} \cdot 2^{-|S|} = 1$. This implies that the probability that $G[S \cup T]$ is nonbipartite is at least $1/2$.

How tight is the above analysis? At the first stage, $\tilde{\Omega}(1/\epsilon)$ random vertices are needed indeed to dominate most of the vertices of G of degree at least $\epsilon n/2$. As for the second stage, an example of a complete bipartite subgraph $K_{\frac{\epsilon n}{2}, \frac{n}{2}}$ (for the induced subgraph on U^1 , say) shows that $\tilde{\Omega}(1/\epsilon^2)$ random vertices are necessary to catch one of its edges with probability $1 - 2^{-\tilde{\Omega}(1/\epsilon)}$. Note, however, that the subgraph $K_{\frac{\epsilon n}{2}, \frac{n}{2}}$ has $\epsilon n/2$ vertices of degree $n/2$. As this degree is much larger than $\epsilon n/2$, we need to sample only $O(1)$ vertices to dominate most of those high degree vertices. Thus, in this case the set S of the first stage does not need to be that large. This in turn reduces the number of partitions of S and makes the requirement for the success probability for a fixed partition of S much less severe.

Our idea will be to represent the first random subset S of size $|S| = \tilde{O}(1/\epsilon)$ as a union of several subsets $S = S_1 \cup \dots \cup S_t$ with $t = \tilde{O}(\ln(1/\epsilon))$, where each S_i dominates most of the vertices of G of degree about n/e^i . (We denote this set by U_i .) Each partition $S = S^1 \cup S^2$ then induces partitions of the subsets: $S_i = S_i^1 \cup S_i^2$ and corresponding partitions $U_i = U_i^1 \cup U_i^2$ of the dominated subsets U_i of $V(G)$. Then if G is an ϵ -robustly nonbipartite graph on n vertices, for each partition $S = S^1 \cup S^2$, one of the sets $U_i^l, l = 1, 2$, will span $\tilde{\Omega}(\epsilon n^2)$ edges. Catching any of them will provide a desired witness for this partition of S . As all degrees in U_i^l are at most n/e^i , this will allow us to apply the so-called generalized Janson inequality to show that if $|T| = \tilde{O}(1/\epsilon)$, then T catches one of the edges inside U_i^l with probability at least $1 - O(2^{-|S_i^l|})$. Then applying the union bound will prove the desired result.

The actual proof will deviate somewhat from the above outline as we will need to overcome some further complications.

In the course of the proof we will need the following lemma.

LEMMA 3.1. *Let $G = (V, E)$ be a graph on n vertices and let $0 < \delta_2 < \delta_1 < 1/2$ be constants. Suppose A, B are disjoint subsets of V . Then with probability at least $1/2$ a random subset $R \subset V$ of size $|R| = (6/\delta_2) \ln^2(1/\delta_1)$ contains a subset $T \subset A$ having the following properties:*

1. $|T| \leq \frac{1}{\delta_1}$.
1. Denote

$$(3.1) \quad B^* = \{v \in B : N(v) \cap T = \emptyset\} .$$

Then

$$(3.2) \quad \sum_{v \in A: d(v, B^*) > \delta_1 n} d(v, B^*) \leq \delta_2 n^2 .$$

The lemma asserts the existence of a set T such that if we remove T from A and the neighbors of T from B most vertex degrees from A to B will be bounded from above by $\delta_1 n$.

Proof of Lemma 3.1. We will generate a random subset R in several steps, each time choosing a random subset R_i of V , where the cardinality of R_i may vary from step to step. At each step we will update the value of T until we will reach T with the desired properties. Then R will be a union of all chosen random subsets R_i .

Denote $s = \ln(1/\delta_1)$. Initially we set $T = \emptyset$, $i = 1$. Define B^* by (3.1). As long as condition (3.2) is not satisfied we do the following. For $1 \leq j \leq s$ define $A_j = \{v \in A : e^{j-1} \delta_1 n < d(v, B^*) \leq e^j \delta_1 n\}$. If for all $1 \leq j \leq s$ one has $|A_j| < \frac{\delta_2 n}{e^j \delta_1 s}$, then

$$\begin{aligned} \sum_{v \in A: d(v, B^*) > \delta_1 n} d(v, B^*) &= \sum_{j=1}^s \sum_{v \in A_j} d(v, B^*) \\ &\leq \sum_{j=1}^s \frac{\delta_2 n}{e^j \delta_1 s} \cdot e^j \delta_1 n \\ &= \delta_2 n^2, \end{aligned}$$

a contradiction. Therefore, there exists an index $1 \leq j_0 = j_0(i) \leq s$ for which $|A_{j_0}| \geq \frac{\delta_2 n}{e^{j_0} \delta_1 s}$. Choose a random subset $R_i \subset V$ of size $|R_i| = (e^{j_0} \delta_1 s / \delta_2) \ln(2/\delta_1)$. The probability that R_i does not intersect A_{j_0} is

$$Pr[R_i \cap A_{j_0} = \emptyset] = \frac{\binom{n - |A_{j_0}|}{|R_i|}}{\binom{n}{|R_i|}} \leq \left(1 - \frac{|A_{j_0}|}{n}\right)^{|R_i|} \leq e^{-\frac{|A_{j_0}| |R_i|}{n}} \leq \frac{\delta_1}{2} .$$

We call step i successful if $R_i \cap A_{j_0} \neq \emptyset$. In this case we choose an arbitrary vertex $v_i \in R_i \cap A_{j_0}$ and denote $d_i = d(v_i, B^*)$. Note that $d_i > e^{j_0-1} \delta_1 n$, implying $|R_i|/d_i \leq (e s / (\delta_2 n)) \ln(2/\delta_1)$. We then add v_i to T , update B^* according to (3.1), and repeat the above described procedure.

Note that after a successful step has been performed, the size of B^* is decreased by at least $\delta_1 n$. Hence at most $1/\delta_1$ successful steps were executed. Consider the event where all steps were successful until the end of the above described iterative procedure. The probability of this event is at least $1 - (1/\delta_1)(\delta_1/2) = 1/2$. As the size of T is equal to the number of successful steps, we get $|T| \leq 1/\delta_1$.

Now define $R = \bigcup_{i=1}^{|T|} R_i$. As $\sum_{i=1}^{|T|} d_i \leq |B| \leq n$, the size of R can be bounded by

$$\begin{aligned} |R| &= \sum_{i=1}^{|T|} |R_i| \leq \sum_{i=1}^{|T|} \frac{es}{\delta_2 n} \ln\left(\frac{2}{\delta_1}\right) d_i \leq \frac{es}{\delta_2} \ln\left(\frac{2}{\delta_1}\right) \leq \frac{e}{\delta_2} \ln\left(\frac{1}{\delta_1}\right) \ln\left(\frac{2}{\delta_1}\right) \\ &< \frac{6}{\delta_2} \ln^2\left(\frac{1}{\delta_1}\right) . \quad \square \end{aligned}$$

Now we briefly outline the proof of Theorem 2. A random set R of size $|R| = 34 \ln^4(1/\epsilon) \ln \ln(1/\epsilon)/\epsilon$ will be generated in three stages, with each stage producing its own set of random vertices R^j , $j = 1, 2, 3$. At the first stage we construct inside R^1 a family of sets $\{S_i\}$, where each S_i has size about $e^i \ln(1/\epsilon)$ and dominates most of the vertices of G with degrees about n/e^i . We denote by U_i the set of vertices of G of degree about n/e^i , dominated by S_i . Note that U_i is not a subset of R^1 ; in fact, with high probability most of U_i will be outside R^1 . At the second stage we use R^2 to adjust the families $\{S_i\}, \{U_i\}$ in such a way that each S_i still dominates U_i , and for each U_i almost all vertices of $\bigcup_{j=1}^{i-1} U_j$ have their degrees into U_i bounded from above by n/e^i . This is a crucial stage which enables us to complete the union of S_i to a nonbipartite subgraph by choosing a random subset R^3 at the third stage.

Let us now introduce some notation. From now until the end of the section we assume that $G = (V, E)$ is an ϵ -robustly non-2-colorable graph on n vertices. Let

$$t = \ln\left(\frac{1}{\epsilon}\right) .$$

Also let, for each $1 \leq i \leq t + 2$,

$$I_i = \left(\frac{n}{e^i}, \frac{n}{e^{i-1}}\right] .$$

Stage 1: Defining S_i 's, U_i 's.

PROPOSITION 3.1. *With probability at least 5/6 a random subset R^1 of V of size $|R^1| = 55t/\epsilon$ contains $t + 2$ disjoint subsets S_1, \dots, S_{t+2} of cardinalities $|S_i| = e^{i+1}t$, $i = 1, \dots, t + 2$, so that for each $1 \leq i \leq t + 2$ the number of vertices of G with degrees in I_i , not dominated by S_i , does not exceed $\frac{\epsilon n}{4(t+2)}$.*

Proof. For each $1 \leq i \leq t + 2$ we choose a subset $S_i \subset V$ of size $|S_i| = e^{i+1}t$ uniformly at random and then take R^1 to be the union of the sets S_i . Note that with probability $1 - o(1)$ the sets S_i are pairwise disjoint. Also,

$$\sum_{i=1}^{t+2} |S_i| = \sum_{i=1}^{t+2} e^{i+1}t \leq te^{t+4} = e^4 \ln\left(\frac{1}{\epsilon}\right) e^{\ln(\frac{1}{\epsilon})} < \frac{55 \ln(\frac{1}{\epsilon})}{\epsilon} .$$

Let X_i be a random variable, counting the number of vertices of G with degrees in I_i , not dominated by S_i . If $v \in V$ has its degree in I_i , then the probability that v is not dominated by S_i can be estimated from above by

$$\frac{\binom{n-d(v)}{|S_i|}}{\binom{n}{|S_i|}} < \left(1 - \frac{d(v)}{n}\right)^{|S_i|} < e^{-\frac{|S_i|n}{e^i n}} = e^{-\frac{e^{i+1}t}{e^i}} = e^{-et} = \epsilon^e .$$

By linearity of expectation we get

$$E[X_i] < n\epsilon^e < \frac{\epsilon n}{80(t+2)^2} .$$

By the Markov inequality $\Pr[X_i > \frac{\epsilon n}{4(t+2)}] < 1/(20(t+2))$. Therefore, the probability that the family $\{S_i\}_{i=1}^{t+2}$ does not satisfy the claim of the lemma is less than $(t+2)\frac{1}{20(t+2)} + o(1) < 1/6$. \square

Now suppose that the first stage is successful and the family $\{S_i\}_{i=1}^{t+2}$ has the property described in the above proposition. For $1 \leq i \leq t+2$ we define

$$U_i = \{v \in V : d(v) \in I_i, N(v) \cap S_i \neq \emptyset\}.$$

It follows from Proposition 3.1 that

$$\begin{aligned} \sum_{v \notin \bigcup_{i=1}^{t+2} U_i} d(v) &\leq \sum_{i=1}^{t+2} \sum_{v \in V: d(v) \in I_i, N(v) \cap S_i = \emptyset} d(v) + \sum_{v \in V: d(v) \leq n/e^{t+2}} d(v) \\ &\leq \sum_{i=1}^{t+2} \frac{\epsilon n}{4(t+2)} \cdot \frac{n}{e^{i-1}} + n \cdot \frac{\epsilon n}{e^2} \\ &< \frac{\epsilon n^2}{2(t+2)} + \frac{\epsilon n^2}{e^2} \\ &< \frac{\epsilon n^2}{2}. \end{aligned}$$

Stage 2: Adjusting S_i 's, U_i 's. The purpose of this stage is to achieve the situation in which for all $2 \leq i \leq t+2$ most of the degrees of vertices from $\bigcup_{j=1}^{i-1} U_j$ to U_i are bounded from above by n/e^{i-1} . We also want S_i to dominate U_i and the size of S_i to remain basically unchanged. Our main technical tool is Lemma 3.1.

For $i = t+2$ down to 2 we repeat the following procedure. Denote $A = U_1 \cup \dots \cup U_{i-1}$, $B = U_i$, $\delta_1 = 1/e^{i-1}$, $\delta_2 = \epsilon/(8(t+2))$. Applying Lemma 3.1 $2 \ln t$ times we get that with probability at least $1 - 1/(6(t+1))$ a random subset $R_i^2 \subset V$ of size $|R_i^2| = 12 \ln t \ln^2(1/\delta_1)/\delta_2 = 96(t+2) \ln t (i-1)^2/\epsilon$ contains a subset T_i of size $|T_i| = e^{i-1}$ having property (3.2) with A , B , δ_1 , and δ_2 as defined above.

Now we update

$$\begin{aligned} S_{i-1} &:= S_{i-1} \cup T_i, \\ U_{i-1} &:= U_{i-1} \cup \{v \in U_i : N(v) \cap T_i \neq \emptyset\}, \\ U_i &:= U_i \setminus U_{i-1}. \end{aligned}$$

PROPOSITION 3.2. *After having executed the above loop, with probability at least $5/6$, the families $\{S_i\}_{i=1}^{t+2}$, $\{U_i\}_{i=1}^{t+2}$ have the following properties:*

1. For every $2 \leq i \leq t+2$

$$(3.3) \quad \sum_{v \in \bigcup_{j=1}^{i-1} U_j, d(v, U_i) > \frac{n}{e^{i-1}}} d(v, U_i) \leq \frac{\epsilon n^2}{8(t+2)}.$$

2. For every $1 \leq i \leq t+2$

$$(3.4) \quad |S_i| \leq t e^{i+2}.$$

3. For every $1 \leq i \leq t+2$ and for every vertex $v \in U_i$,

$$(3.5) \quad d(v) \leq \frac{n}{e^{i-1}}.$$

4. *Still*

$$(3.6) \quad \sum_{v \notin \bigcup_{i=1}^{t+2} U_i} d(v) \leq \frac{\epsilon n^2}{2}.$$

Proof. Note that before starting stage 2, all vertices in U_i have their degrees bounded from above by n/e^{i-1} . Therefore, moving some of them to U_{i-1} cannot create vertices $v \in \bigcup_{j=1}^{i-1} U_j$ for which $d(v, U_i) > n/e^{i-1}$. Also, as we proceed downwards from $i = t + 2$ to $i = 2$, once we have moved vertices from U_i to U_{i-1} , the set U_i remains unchanged. Therefore, (3.3) follows from Lemma 3.1. Similarly, (3.4) follows from the estimate $|S_i| \leq te^{i+1}$ before the execution of stage 2 and the fact $|T_{i+1}| = e^i$. Note that the new U_i is a subset of the union of the old U_j , $j = i, \dots, t + 2$. As before stage 2 we have $d(v) \leq n/e^{i-1}$ for all $v \in \bigcup_{j=i}^{t+2} U_j$, (3.5) follows. Finally, as the union $\bigcup_{i=1}^{t+2} U_i$ remains the same after stage 2, (3.6) follows from the corresponding property of the old sets U_i . \square

Let $R^2 = \bigcup_{i=2}^{t+2} R_i^2$ be the random vertices consumed at stage 2. We have

$$|R^2| = \sum_{i=2}^{t+2} |R_i^2| = \sum_{i=2}^{t+2} \frac{96(t+2) \ln t(i-1)^2}{\epsilon} < \frac{33t^4 \ln t}{\epsilon}.$$

Stage 3: Completing $\bigcup_{i=1}^{t+2} S_i$ to a nonbipartite subgraph. Assume now that the graph G on n vertices is ϵ -far from being bipartite. Our aim is to show that with probability at least $11/12$ the union of $\bigcup_{i=1}^{t+2} S_i$, with S_i as defined in the end of stage 2, and a random subset $R^3 \subset V$ of an appropriately chosen size forms a nonbipartite subgraph of G . This will follow easily from the proposition below.

PROPOSITION 3.3. *Let $G = (V, E)$ be an ϵ -robustly nonbipartite graph on n vertices. Let the subsets $\{S_i\}_{i=1}^{t+2}$, $\{U_i\}_{i=1}^{t+2}$ satisfy (3.3)–(3.6). Denote $S = \bigcup_{i=1}^{t+2} S_i$. Then with probability at least $5/6$ a random subset R^3 of size $|R^3| = 2700t^2/\epsilon$ has the following property. For every partition $S = S^1 \cup S^2$ of S there exists an edge $e = (u, v) \in E(G)$ with $u, v \in R^3$ and both u, v having neighbors in the same S^l for some $l \in \{1, 2\}$.*

Proof. For a fixed partition $S = S^1 \cup S^2$ we denote, for $1 \leq i \leq t + 2$, $l = 1, 2$, $S_i^l = S^l \cap S_i$. We also set $U_i^1 = \{v \in U_i : N(v) \cap S_i^2 \neq \emptyset\}$, $U_i^2 = U_i \setminus U_i^1$. Let G_i^l be the following graph. The vertex set of G_i^l is $\bigcup_{j=1}^i U_j^l$; an edge $e = (u, v) \in E(G)$ is an edge of G_i^l if and only if $u, v \in U_i^l$ or $u \in \bigcup_{j=1}^{i-1} U_j^l$, $v \in U_i^l$, and $d(u, U_i^l) \leq n/e^{i-1}$. Note that by (3.5) all degrees in G_i^l are at most n/e^{i-1} .

As the graph G is at least ϵn^2 edges far from any bipartite graph, we get, recalling (3.6), that either U^1 or U^2 span at least $\epsilon n^2/4$ edges. Therefore, for some $1 \leq i \leq t + 2$, $l \in \{1, 2\}$ we have

$$e \left(\bigcup_{j=1}^{i-1} U_j^l, U_i^l \right) + e(U_i^l) \geq \frac{\epsilon n^2}{4(t+2)}.$$

A partition (S^1, S^2) of S is called (i, l) -bad, if $|E(G_i^l)| \geq \epsilon n^2/(8(t+2))$ and $|E(G_j^{l'})| < \epsilon n^2/(8(t+2))$, for all $j < i$, $l' \in \{1, 2\}$. From the definition of G_i^l we get, using (3.3), that any partition (S^1, S^2) is (j, l) -bad for some $1 \leq j \leq t + 2$, $l \in \{1, 2\}$.

Two (j, l) -bad partitions (S^1, S^2) , $((S^1)', (S^2)')$ are called *equivalent* if $S_i^1 = (S_i^1)'$ for $1 \leq i \leq j$ (and thus $U_i^1 = (U_i^1)'$). By (3.4) for a fixed $1 \leq j \leq t + 2$, the total

number of equivalence classes of (j, l) -bad partitions, where $l \in \{1, 2\}$, is at most

$$2 \cdot 2^{\sum_{i=1}^j |S_i|} \leq 2^{1+\sum_{i=1}^j e^{i+2}t} \leq e^{e^{j+3}t} .$$

Note crucially that two (j, l) -bad partitions in the same equivalence class have the same graph G_j^l . It follows easily from this observation that it is enough to prove that with probability at least $5/6$ the random subset R^3 spans an edge of G_j^l for every $1 \leq j \leq t + 2$, every $l \in \{1, 2\}$, and every equivalence class of (j, l) -bad partitions.

In this proof it is convenient to generate R^3 by choosing each vertex $v \in V$ independently with probability $p = 2700t^2/\epsilon n$. This will allow us to use the so-called generalized Janson inequality (see, e.g., [1, Ch. 8]) to estimate the probability that R^3 misses all edges of G_j^l for some fixed equivalence class of (j, l) -bad partitions.

Consider some fixed equivalence class of (j, l) -bad partitions and its graph G_j^l . Note that $|E(G_j^l)| \geq \epsilon n^2/(8(t+2))$ and also that the maximal degree of G_j^l is bounded from above by n/e^{j-1} . Denote by Y the random variable counting the number of edges of G_j^l , spanned by R^3 . Then $E[Y] = |E(G_j^l)|p^2$. Our aim is to estimate from above the probability that R^3 spans no edges of G_j^l , i.e., $Pr[Y = 0]$. A naive analysis performed by choosing the vertices of R^3 pair after pair and requiring that each pair does not coincide with an edge of G_j^l gives only $Pr[Y = 0] \leq (1 - |E(G_j^l)|/\binom{n}{2})^{|R^3|/2}$. We will get a better estimate using the assumption on the maximal degree in G_j^l . Let

$$\Delta = 2 \sum_{\substack{e \neq e' \in E(G_j^l) \\ e \cap e' \neq \emptyset}} Pr[e, e' \subset R^3] .$$

Then

$$\begin{aligned} \Delta &= \sum_{e \in E(G_j^l)} \sum_{\substack{e \neq e' \in E(G_j^l) \\ e' \cap e \neq \emptyset}} Pr[e, e' \subset R^3] \\ &= \sum_{e=(u,v) \in E(G_j^l)} ((d_{G_j^l}(u) - 1) + (d_{G_j^l}(v) - 1))p^3 \\ &< \sum_{e \in E(G_j^l)} \frac{2np^3}{e^{j-1}} = \frac{2|E(G_j^l)|np^3}{e^{j-1}} . \end{aligned}$$

By the generalized Janson inequality,

$$Pr[Y = 0] \leq e^{-\frac{(E[Y])^2}{3\Delta}} = e^{-\frac{|E(G_j^l)|e^{j-1}p}{6n}} \leq e^{-\frac{\epsilon n e^{j-1} p}{48(t+2)}} < \frac{e^{-e^{j+3}t}}{6(t+2)} .$$

Recalling the estimate on the number of equivalence classes of (j, l) -bad partitions, we conclude that the probability that R^3 does not contain an edge of G_j^l for some equivalence class is at most

$$\sum_{j=1}^{t+2} e^{e^{j+3}t} \frac{e^{-e^{j+3}t}}{6(t+2)} = 1/6 . \quad \square$$

Now assume that stages 1 and 2 were successful and the set R^3 has the property stated in Proposition 3.3. Then it is easy to see that the spanned subgraph $G[S \cup R^3]$

is not bipartite. Indeed, let $c : S \cup R^3 \rightarrow \{1, 2\}$ be a 2-coloring of $S \cup R$. Define a partition $S = S^1 \cup S^2$ of S by $S^1 = \{v \in S : c(v) = 1\}$, $S^2 = \{v \in S : c(v) = 2\}$. Then R^3 contains an edge $e = (u, v) \in E(G)$ with both endpoints u, v connected to one color class, say S^1 . If c colors u or v in color 1, we get a monochromatic edge connecting u or v , respectively, with S^1 . Otherwise, $c(u) = c(v) = 2$, but then e is monochromatic under c . By the above analysis with probability at least $2/3$ the random sets R^1 and R^2 define a subset $S \subset R^1 \cup R^2$ with the properties stated in Proposition 3.2. Therefore, with probability at least $1/2$ the union $S \cup R^3$ spans a nonbipartite subgraph of G .

It remains only to estimate the size of the random set $R = R^1 \cup R^2 \cup R^3$. We have

$$|R| = |R^1| + |R^2| + |R^3| = \frac{55t}{\epsilon} + \frac{33t^4 \ln t}{\epsilon} + \frac{2700t^2}{\epsilon} < \frac{34 \ln^4 \left(\frac{1}{\epsilon}\right) \ln \ln \left(\frac{1}{\epsilon}\right)}{\epsilon} .$$

The proof of Theorem 2 is complete. \square

4. Testing k -colorability. In this section we prove Theorem 3. It will be convenient to generate a random subset $R \subset V(G)$ of size $|R| = s = 36k \ln k / \epsilon^2$ in s rounds, each time choosing uniformly at random a single vertex $r_j \in V(G)$. This in principle may result in choosing one vertex several times and thus getting a set of cardinality less than s . However, the probability of this event is $o(1)$, and therefore the approach for generating R we take here is asymptotically equivalent to choosing a subset of V of size s uniformly at random.

Our basic approach is similar to the one of Goldreich, Goldwasser, and Ron [3]. At the end of the section we explain the main differences and the reason our argument saves a factor of $\Theta(1/\epsilon)$ in the number of vertices sampled.

Let G be an ϵ -robustly non- k -colorable graph on n vertices. Suppose we are given a subset $S \subset V(G)$ (of the sample set R) and its k -partition $\phi : S \rightarrow [k]$; our aim is to find with high probability inside the next several random vertices a succinct witness to the fact that ϕ cannot be extended to a proper coloring of the sample. If a k -coloring $c : V(G) \rightarrow [k]$ of G is to coincide with ϕ on S , then for every vertex $v \in V \setminus S$ the colors of neighbors of v in S under ϕ are forbidden for v in c . The rest of the colors are still feasible for v . It could be that v has no feasible colors left at all. Such a vertex will be called colorless with respect to S and ϕ . If the number of colorless vertices is large, then there is a decent chance that between the next few random vertices of R there will be one such colorless vertex v^* . Obviously, adding v^* to S provides the desired witness for nonextendibility of ϕ .

If the set of colorless vertices is small, then one can show that, as G is ϵ -far from being k -colorable, there is a relatively large subset W of vertices (which will be called restricting) such that adding any vertex $v \in W$ to S and coloring it by any feasible color with regard to ϕ excludes this color from the lists of feasible colors of at least $\Omega(\epsilon)n$ neighbors of v . If such v is caught in the next few vertices of the random sample R , then adding v to S and coloring it by any of its feasible colors reduces substantially the total length of the lists of feasible colors for the vertices of V , thus helping to approach the first situation, i.e., the case when there are many colorless vertices. As the reader can guess, the above described process can be represented by a tree in which every node corresponds to a colorless or restricting vertex v and each edge corresponds to a feasible color for v . As the degree of such a node can be as large as k , the size of the tree grows quickly as we proceed with choosing vertices from R and can reach size exponential in $1/\epsilon$. We therefore will need the probability

of success (i.e., the probability of catching a colorless/restricting vertex) along several consecutive steps to be exponentially close to 1.

Now we present a formal description of the above argument. First we need to introduce some notation. We denote the set $\{1, \dots, k\}$ by $[k]$. Suppose $G = (V, E)$ is a graph on n vertices. Given a subset $S \subseteq V$ and its k -partition $\phi : S \rightarrow [k]$, for every $v \in V \setminus S$ let

$$L_\phi(v) = [k] \setminus \{1 \leq i \leq k : \exists u \in S \cap N(v), \phi(u) = i\} .$$

If $S = \emptyset$, we set $L_\phi(v) = [k]$ for every $v \in V$. If a k -coloring $c : V \rightarrow [k]$ of G coincides with ϕ on S , then for every $v \in V \setminus S$ the color of v in c belongs to $L(v)$. For this reason, the set $L_\phi(v)$ is called the *list of feasible colors* for v . A vertex $v \in V \setminus S$ is called *colorless* if $L_\phi(v) = \emptyset$. We denote by U the set of all colorless vertices under (S, ϕ) .

For every vertex $v \in V \setminus (S \cup U)$ define

$$\delta_\phi(v) = \min_{i \in L_\phi(v)} |\{u \in N(v) \setminus (S \cup U) : i \in L_\phi(u)\}| .$$

Thus coloring v by one of the colors from $L_\phi(v)$ and then adding it to S results in deleting this color and thus shortening the lists of feasible colors of at least $\delta_\phi(v)$ neighbors of v outside S .

CLAIM 4.1. *For every set $S \subset V$ and every k -partition ϕ of S , the graph G is at most $(n - 1)|S \cup U| + \sum_{v \in V \setminus (S \cup U)} \delta_\phi(v)$ edges far from being k -colorable.*

Proof. For every $v \in S$, color v according to $\phi(v)$. For every $v \in U$ we color v in an arbitrary color from $[k]$. For every $v \in V \setminus (S \cup U)$ we color v in color $i \in L_\phi(v)$ for which $\delta_\phi(v) = |\{u \in N(v) \setminus (S \cup U) : i \in L_\phi(u)\}|$. Let us estimate the number of monochromatic edges under this coloring. The number of monochromatic edges incident with $S \cup U$ is at most $(n - 1)|S \cup U|$. Every vertex $v \in V \setminus (S \cup U)$ has exactly $\delta_\phi(v)$ neighbors $u \in V \setminus (S \cup U)$, whose color list $L_\phi(v)$ contains the color chosen for v . Therefore, v will have at most $\delta_\phi(v)$ neighbors in $V \setminus (S \cup U)$ colored in the same color. Hence the total number of monochromatic edges is at most $(n - 1)|S \cup U| + \sum_{v \in V \setminus (S \cup U)} \delta_\phi(v)$, as claimed. \square

COROLLARY 4.1. *If G is an ϵ -robustly non- k -colorable graph on n vertices, then for any pair (S, ϕ) , where $S \subset V(G)$, $\phi : S \rightarrow [k]$, one has*

$$\sum_{v \in V \setminus (S \cup U)} \delta_\phi(v) > \epsilon n^2 - n(|S| + |U|) ,$$

where U is the set of colorless vertices for the pair (S, ϕ) .

Given a pair (S, ϕ) , a vertex $v \in V \setminus (S \cup U)$ is called *restricting* if $\delta_\phi(v) \geq \epsilon n/2$. We denote by W the set of all restricting vertices.

CLAIM 4.2. *If G is an ϵ -robustly non- k -colorable graph on n vertices, then for every pair (S, ϕ) , where $S \subset V(G)$ and $\phi : S \rightarrow [k]$, one has*

$$|U \cup W| > \frac{\epsilon n}{2} - |S| .$$

Proof. By Corollary 4.1,

$$\epsilon n^2 - n(|S| + |U|) < \sum_{v \in V \setminus (S \cup U)} \delta_\phi(v) \leq |W|(n - 1) + \sum_{v \in V \setminus (S \cup U \cup W)} \delta_\phi(v) < |W|n + \frac{n \cdot \epsilon n}{2} .$$

This implies $|S| + |U| + |W| \geq \epsilon n/2$. As U and W are disjoint, the result follows. \square

Now let G be an ϵ -robustly non- k -colorable graph on n vertices. While choosing random vertices r_1, \dots, r_s of R we construct an auxiliary k -ary tree T . To distinguish between the vertices of G and those of T we call the latter *nodes*. Each node of T is labeled either by a vertex of G or by the special symbol $\#$, whose meaning will be explained soon. If a node t of T is labeled by $\#$, then t is called a *terminal node*. The edges of T are labeled by integers from $[k]$.

Let t be a node of T . Consider the path from the root of T to t , not including t itself. The labels of the nodes along this path form a subset $S(t)$ of $V(G)$. The labels of the edges along the path define a k -partition $\phi(t)$ of $S(t)$ in the natural way: the label of the edge following a node t' in the path determines the color of its label $v(t')$. The labeling of the nodes and edges of T will have the following property: if t is labeled by v and v has a neighbor in $S(t)$ whose color in $\phi(t)$ is i , then the son of v along the edge labeled by i is labeled by $\#$. This label indicates the fact that in this case color i is infeasible for v , given $(S(t), \phi(t))$.

At each step of the construction of T we will maintain the following: all leafs of T are either unlabeled or are labeled by $\#$. Also, only leafs of T can be labeled by $\#$. We start the construction of T from an unlabeled single node, the root of T .

Suppose that $j - 1$ vertices of T have already been chosen, and we are about to choose vertex r_j of R . Consider a leaf t of T . If t is labeled by $\#$, we do nothing for this leaf. (That is the reason such a t is called a terminal node; nothing will ever grow out of this node.) Now assume that t is unlabeled. Define the pair $(S(t), \phi(t))$ as described above. Now, for the pair $(S(t), \phi(t))$ we define the set $U(t)$ of colorless vertices and the set $W(t)$ of restricting vertices as described before. Round j is called *successful* for the node t if the random vertex r_j satisfies $r_j \in U(t) \cup W(t)$. If round j is indeed successful for t , then we label t by r_j , create k sons of t , and label the corresponding edges by $1, \dots, k$. Now, if color i is infeasible for r_j , given $(S(t), \phi(t))$, we label the son of t along the edge with label i by $\#$; otherwise we leave this son unlabeled. Note that if $r_j \in U(t)$, then none of the colors from $[k]$ is feasible for r_j , and thus all the sons of t will be labeled by $\#$. This completes the description of the process of constructing T .

Now we state some properties of T .

CLAIM 4.3. *The depth of T is bounded from above by $\frac{2k}{\epsilon}$.*

Proof. Let t^* be a leaf of T . Notice that if the label of a node t of T belongs to $U(t)$, then all sons of t in T are labeled by $\#$ and are terminal nodes. Therefore, all nodes on the path from the root of T to t^* , but possibly the node immediately preceding t^* , have their labels in the corresponding sets $W(t)$. Since each vertex in $W(t)$ is restricting with respect to $(S(t), \phi(t))$, coloring v in any feasible color decreases the total size of the lists of feasible colors for all vertices of G by at least $\epsilon n/2$. Therefore, each time when on the path from the root of T to t^* we leave a node t , whose label belongs to $W(t)$, the total length of the lists of feasible colors shrinks by at least $\epsilon n/2$. As initially all k colors are feasible for all vertices, we start with lists of feasible colors of total length nk . Thus we cannot make more than $nk/(\epsilon n/2) = 2k/\epsilon$ steps down from the root of T to t^* . This implies that the depth of T is at most $2k/\epsilon$. \square

CLAIM 4.4. *If a leaf t^* of T is labeled by $\#$, then $\phi(t^*)$ is not a proper k -coloring of $S(t^*)$.*

Proof. By the definition of the labeling procedure, let t' be the father of t^* in T .

Let v be the label of t' , and let i be the label of the edge of T connecting t' and t^* . Since t^* is labeled by “#,” i is not a feasible color for v , given $(S(t'), \phi(t'))$. As $\phi(t^*)$ colors v in color i , we get the existence of an edge spanned by $S(t^*)$, incident with v and monochromatic under $\phi(t^*)$. \square

CLAIM 4.5. *If after round j all leafs of the tree T are terminal nodes, then the subgraph $G[\{r_1, \dots, r_j\}]$ is not k -colorable.*

Proof. Notice first that the labels of all nodes of T are either # or vertices from $\{r_1, \dots, r_j\}$. Let $c : \{r_1, \dots, r_j\} \rightarrow [k]$ be a k -partition of $\{r_1, \dots, r_j\}$. In order to show that c creates some monochromatic edges in the induced subgraph of G on $\{r_1, \dots, r_j\}$, we start with the root t_0 of T and traverse T guided by c as follows: while at a node t of T , labeled by $v(t) \in \{r_1, \dots, r_j\}$, we move from t to its son along the edge of T labeled by $c(v(t))$. Once we reach a terminal node t^* of T , we then have $S(t^*) \subseteq \{r_1, \dots, r_j\}$ and $\phi(t^*)$ coincides with c on $S(t^*)$. As t^* is a terminal node, it follows from Claim 4.4 that c is not a proper k -coloring of $S(t^*)$. \square

CLAIM 4.6. *If G is ϵ -robustly non- k -colorable graph on n vertices, then after $36k \ln k/\epsilon^2$ rounds with probability at least $1/2$ all leaves of T are terminal nodes.*

Proof. As every nonleaf node of T has k sons and by Claim 4.3 T has depth at most $2k/\epsilon$, it can be embedded naturally in the k -ary tree $T_{k, \frac{2k}{\epsilon}}$ of depth $2k/\epsilon$. Moreover, this embedding can be prefixed even before exposing R and T . Note that the number of vertices of $T_{k, \frac{2k}{\epsilon}}$ is $1 + k + \dots + k^{\frac{2k}{\epsilon}} \leq k^{\frac{2k}{\epsilon} + 1}$.

Recall that during the construction of the random sample R and the tree T , a successful round for a leaf t of T results in creating k sons of T . Fix some node t of $T_{k, \frac{2k}{\epsilon}}$. If after $36k \ln k/\epsilon^2$ rounds t is a leaf of T , then the total number of successful rounds for the path from the root of T to t is equal to the depth of t . As $S(t) \subseteq R$ and thus $|S(t)| = O(1)$, by Claim 4.2 each round has probability of success at least $\epsilon/3$. Therefore, the probability that t is a nonterminal leaf of T after $36k \ln k/\epsilon^2$ steps can be bounded from above by the probability that the Binomial random variable $B(36k \ln k/\epsilon^2, \epsilon/3)$ is less than $2k/\epsilon$. The latter probability is at most

$$\exp \left\{ -\frac{\left(\frac{12k \ln k}{\epsilon} - \frac{2k}{\epsilon}\right)^2}{\frac{24k \ln k}{\epsilon}} \right\} < \exp \left\{ -\frac{\left(\frac{9k \ln k}{\epsilon}\right)^2}{\frac{24k \ln k}{\epsilon}} \right\} = e^{-\frac{27k \ln k}{8\epsilon}} < k^{-\frac{3k}{\epsilon}}.$$

Thus, by the union bound we conclude that the probability that some node of $T_{\epsilon, \frac{2k}{\epsilon}}$ is a leaf of T , non labeled by “#,” is at most $|(V(T_{k, \frac{2k}{\epsilon}}))|k^{-\frac{3k}{\epsilon}} < \frac{1}{2}$. \square

Proof of Theorem 3. The proof follows immediately from Claims 4.5 and 4.6. \square

Note that our proof here is similar to the basic argument of Goldreich, Goldwasser, and Ron in [3]. They also construct (implicitly) the tree T constructed in the course of our proof. Their argument can be briefly described as follows: given a current tree T , Goldreich, Goldwasser, and Ron require that the next subset R_i of a random sample R contains, with high probability, for every leaf $t \in T$, a vertex $v \in U(t) \cup W(t)$. As each random vertex r_j hits $U(t) \cup W(t)$ with probability at least $\epsilon/3$ by Claim 4.2, the probability that for a fixed $t \in T$ the next $\tilde{O}(1/\epsilon^2)$ random vertices will not hit the set $U(t) \cup W(t)$ is at most $(1 - \epsilon/3)^{\tilde{O}(1/\epsilon^2)} = 2^{-\tilde{O}(1/\epsilon)}$. The number of leafs of T is at most $2^{O(1/\epsilon)}$. Therefore, by the union bound the set $R_j \subset R$ of $|R_j| = \tilde{O}(1/\epsilon^2)$ random vertices hits the set $U(t) \cup W(t)$ for every leaf $t \in T$ with probability $1 - 2^{O(1/\epsilon)}2^{-\tilde{O}(1/\epsilon)} = 1 - o(\epsilon)$. Thus, representing $R = R_1 \cup \dots \cup R_{\frac{2k}{\epsilon}}$ with $|R_j| = \tilde{O}(1/\epsilon^2)$, they ensure that a.s. each time after having chosen the next

piece R_j of random vertices, all nonterminal leaves of T will get k sons each. As by Claim 4.3 the depth of T is bounded by $2k/\epsilon$, after having chosen all $2k/\epsilon$ random pieces $R_1, \dots, R_{\frac{2k}{\epsilon}}$, a.s. all leaves of T are terminal nodes. In contrast, in our proof we require only that along each path in the tree $T_{k, \frac{2k}{\epsilon}}$ sufficiently many steps will be successful, not insisting on the regularity of appearance of successful steps. This results in saving a factor of $\tilde{\Theta}(1/\epsilon)$.

5. Concluding remarks and open problems. As mentioned in the introduction, the study of the function $g_k(n, \epsilon)$ is motivated by its relevance to the design of efficient testing algorithms for k -colorability. Thus, Theorem 2 shows that bipartiteness can be tested by choosing randomly some $\tilde{O}(1/\epsilon)$ random vertices and by checking if the induced subgraph on them is 2-colorable. Here, as usual, $\tilde{O}(1/\epsilon)$ denotes $\frac{(\log(1/\epsilon))^{O(1)}}{\epsilon}$. Moreover, by Theorem 1, part 1, any bipartiteness testing algorithm that checks induced subgraphs and is a one-way error algorithm (that is, never errs on bipartite graphs) must check induced subgraphs on at least $\Omega(1/\epsilon)$ vertices.

Similarly, Theorem 3 provides, for every fixed $k \geq 3$, a one-way error algorithm that tests k -colorability by checking random induced subgraphs on $O(1/\epsilon^2)$ vertices. Both algorithms improve the results in [3].

It will be nice to close the gap between our upper and lower bounds for the functions $g_k(n, \epsilon)$ and $f_k(n, \epsilon)$ for $k \geq 3$. It is plausible to conjecture that for every fixed $k \geq 3$, $g_k(n, \epsilon) = \tilde{O}(1/\epsilon)$ and $f_k(n, \epsilon) = \tilde{O}(1/\epsilon)$. This remains open.

Finally we note that Goldreich, Goldwasser, and Ron measure the complexity of their algorithms for graph property testing by the number of pairs of vertices (u, v) of the input graph G queried by the algorithm. The query complexity of our algorithms for testing k -colorability is $\tilde{O}(1/\epsilon^2)$ for $k = 2$ and $(1/\epsilon^4)$ for $k \geq 3$. It is easy to prove a lower bound of $\Omega(1/\epsilon)$ for testing k -colorability. It would be quite interesting to obtain tighter bounds for the query complexity of this problem.

Acknowledgment. We would like to thank the anonymous referees for their very careful reading of the first version of the paper and for many helpful comments and suggestions.

REFERENCES

- [1] N. ALON AND J. H. SPENCER, *The Probabilistic Method*, Wiley, New York, 1992.
- [2] B. BOLLOBÁS, P. ERDŐS, M. SIMONOVITS, AND E. SZEMERÉDI, *Extremal graphs without large forbidden subgraphs*, Ann. Discrete Math., 3 (1978), pp. 29–41.
- [3] O. GOLDREICH, S. GOLDWASSER, AND D. RON, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750.
- [4] J. KOMLÓS, *Covering odd cycles*, Combinatorica, 17 (1997), pp. 393–400.
- [5] V. RÖDL AND R. DUKE, *On graphs with small subgraphs of large chromatic number*, Graphs Combin., 1 (1985), pp. 91–96.
- [6] E. SZEMERÉDI, *Regular partitions of graphs*, in Proceedings of the Colloque International CNRS, J. C. Bermond, J. C. Fournier, M. Las Vergnas, and D. Sotteau, eds., 1978, CNRS, Paris, pp. 399–401.

MULTICOVERING BOUNDS FROM RELATIVE COVERING RADII*

IIRO HONKALA[†] AND ANDREW KLAPPER[‡]

Abstract. The multicovering radii of a code are recently introduced natural generalizations of the covering radius measuring the smallest radius of balls around codewords that cover all m -tuples of vectors. In this paper we prove a new identity relating the multicovering radii of a code to a relativized notion of ordinary covering radius. This identity is used to prove new bounds on the multicovering radii of particular codes.

Key words. covering radius, multicovering radius, coding theory, Hamming codes

AMS subject classifications. 94B65, 94B75, 94B05, 05B40

PII. S089548010037935X

1. Introduction and definitions. The concept of multicovering radius was introduced by Klapper [5] in the context of studying the existence of stream ciphers secure against a large class of attacks. Let C be a code of length n and m be a positive integer. The m -covering radius of C is the smallest integer r such that every set of m vectors in \mathbf{F}^n is contained in at least one ball of radius r around a codeword in C .

We denote the m -covering radius of a code C by $R_m(C)$. Then $R(C) := R_1(C)$ is the covering radius of C . For results on the covering radius, we refer to the book by Cohen et al. [1]. For earlier results on multicovering radii, see [4, 5, 6].

In general we are interested in various extremal values associated with this notion:

- $t_m(n) = R_m(\mathbf{F}^n) =$ the smallest m -covering radius among length n codes.
- $t_m(n, K) =$ the smallest m -covering radius among (n, K) codes, i.e., codes of length n with cardinality K .
- $K_m(n, R) =$ the smallest cardinality of a length n code with m -covering radius R .
- $\ell_m(a, R) =$ the smallest length of a linear code with codimension a and m -covering radius R .

When $m = 1$ we sometimes omit the subscript m . As usual, when we are concerned only with linear codes, parentheses are replaced by square brackets and the size K is replaced by the dimension k . As with the classical covering radius, a variety of bounds are known for these quantities [5, 6], but precise values are known only in a few cases.

There are, of course, relationships among these values. These can be proved by straightforward generalizations of the arguments used by Cohen et al. [1].

LEMMA 1.1. *For positive integers m, n, R, a, k , and n_0 we have the following:*

1. *If $\ell_m(a, R) \leq n_0$ and $n \geq n_0$, then $t_m(n, 2^{n-a}) \leq R$.*

*Received by the editors October 12, 2000; accepted for publication (in revised form) January 2, 2002; published electronically March 20, 2002. The paper was presented in the IEEE International Symposium on Information Theory, Sorrento, Italy, 2000.

<http://www.siam.org/journals/sidma/15-2/37935.html>

[†]Department of Mathematics, University of Turku, 20014 Turku, Finland (honkala@utu.fi). This author's research was supported by the Academy of Finland under grant 44002.

[‡]Department of Computer Science, 763H Anderson Hall, University of Kentucky, Lexington, KY 40506-0046 (klapper@cs.uky.edu). This author's research was supported by the National Science Foundation under grant NCR-9706078.

- 2. If $K_m(n, R) \leq K \leq 2^n$, then $t_m(n, K) \leq R$.
- 3. If $K_m(n, R) > K$, then $t_m(n, K) > R$.

The purpose of this paper is to derive new bounds by relating the multicovering radii of a code to a relativized notion of covering radius. We obtain the following results: new upper bounds and, in some cases, precise values for $R_m(\mathbf{F}^n)$; precise values for $R_3(\mathcal{H}_r)$, where \mathcal{H}_r is the Hamming code of degree r ; lower bounds for $R_m(C)$ for certain values of m ; and an upper bound on $R_m(C)$ in terms of the minimum distance of C .

For generality, we define the notion of relativized covering radius for multicovering radii, although we use only the ordinary covering radius version in this paper.

DEFINITION 1.2. *Let C and S be codes of length n , and let m be a positive integer. Then the m -covering radius of S relative to C , $R_m(S, C)$ is the smallest integer r such that for every $c^1, \dots, c^m \in C$ there is an $x \in S$ such that $d(c^i, x) \leq r$ for all $i = 1, \dots, m$. We also let $t_m(s, C) = \min\{R_m(S, C) : |S| = s\}$.*

Note that $R_m(S, \mathbf{F}^n) = R_m(S)$ and $t_m(s, \mathbf{F}^n) = t_m(n, s)$.

2. A fundamental identity. In this section we prove a new identity relating the m -covering radius of a code C to the covering radii of cardinality m codes relative to C . For any code S , we denote the set of word-complements of elements of S by \bar{S} . (The complement of a word x is $x + 111 \dots 1$ by definition.)

THEOREM 2.1. *Let C be a code of length n . Then*

$$R_m(C) = n - t_1(m, C).$$

Proof. Let S be any (n, m) code. Then

$$(1) \quad R_1(\bar{S}, C) \geq t_1(m, C),$$

with equality for at least one such S . Therefore there is some $c \in C$ such that for every $x \in \bar{S}$, $d(c, x) \geq t_1(m, C)$. This is the same as saying that there is some $c \in C$ such that for every $x \in S$, $d(c, x) \leq n - t_1(m, C)$. Since this holds for every (n, m) code S , we have $R_m(C) \leq n - t_1(m, C)$.

If equality holds in (1), then for every $c \in C$ there is an $x \in \bar{S}$ such that $d(c, x) \leq t_1(m, C)$. This is the same as saying that for every $c \in C$ there is an $x \in S$ such that $d(c, x) \geq n - t_1(m, C)$. Thus $R_m(C) \geq n - t_1(m, C)$. Since (1) holds with equality for at least one S , we have $R_m(C) = n - t_1(m, C)$. \square

For $C = \mathbf{F}^n$ we obtain the following corollary, which is essentially a restatement of Theorem 19.4.4 of Cohen et al. [1] (cf. also Theorem 19.4.2).

COROLLARY 2.2. *For all natural numbers $n, m \geq 1$, $t_m(n) = R_m(\mathbf{F}^n) = n - t_1(n, m)$.*

Proof. This follows from Theorem 2.1 with $C = \mathbf{F}^n$ and the fact that $t_1(m, \mathbf{F}^n) = t_1(n, m)$. \square

Thus bounds on $t_1(n, m)$ give bounds on $R_m(\mathbf{F}^n)$. It was previously shown [5] that

$$(2) \quad R_m(\mathbf{F}^n) \geq \frac{n + \lceil \log_2(m) \rceil - 1}{2}$$

for all $m \leq 2^n$. Since $t_1[n, k] \geq t_1(n, 2^k)$, an upper bound on $t_1(n, m)$ or $t_1[n, k]$ also gives us a lower bound on $R_m(C)$ for any length n code C . Furthermore, by Lemma 1.1, a bound of the form $\ell_1(a, R) = \ell(a, R) \leq n_0$ gives a bound $R_{2^{n-a}}(C) \geq n - R$ for any $n \geq n_0$ and any C of length n . Similarly, $K_1(n, R) \leq k$ if and only if $R_k(C) \geq n - R$

TABLE 1
Lower bounds on $R_{2^k}(C)$.

k	$t[n, k]$	$R_{2^k}(C) \geq$	if
1	$\lfloor \frac{n}{2} \rfloor$	$\lceil \frac{n}{2} \rceil$	$n \geq 1$
2	$\lfloor \frac{n-1}{2} \rfloor$	$\lceil \frac{n+1}{2} \rceil$	$n \geq 2$
3	$\lfloor \frac{n-2}{2} \rfloor$	$\lceil \frac{n+2}{2} \rceil$	$n \geq 3$
4	$\lfloor \frac{n-4}{2} \rfloor$	$\lceil \frac{n+4}{2} \rceil$	$n \geq 4, n \neq 5$
5	$\lfloor \frac{n-5}{2} \rfloor$	$\lceil \frac{n+5}{2} \rceil$	$n \geq 5, n \neq 6$
6	$\leq \lfloor \frac{n-8}{2} \rfloor$	$\lceil \frac{n+8}{2} \rceil$	$n \geq 14$
7	$\leq \lfloor \frac{n-9}{2} \rfloor$	$\lceil \frac{n+9}{2} \rceil$	$n \geq 19$
8	$\leq \lfloor \frac{n-16}{2} \rfloor$	$\lceil \frac{n+16}{2} \rceil$	$n \geq 127$
$2p+1$	$\leq \lfloor \frac{n-2^p}{2} \rfloor$	$\lceil \frac{n+2^p}{2} \rceil$	$n \geq 2^{2p}-1$
$2p$	$\leq \lfloor \frac{n-2^{p-1/2}}{2} \rfloor$	$\lceil \frac{n+2^{p-1/2}}{2} \rceil$	$n \geq 2^{2p-1}$

for every code of length n . Many such bounds are known, and they are well surveyed by Cohen et al. [1]. We summarize the implications for the m -covering radius of \mathbf{F}^n in several tables. Table 1 is a corollary of Theorems 5.2.3, 5.2.7, 5.2.10, 5.2.16, and 5.2.21 in [1]. Here C is any code of length n . We also have $t_1[5, 4] = t_1[6, 5] = 1$; so $R_{16}(C) \geq 4$ if C has length 5, and $R_{32}(C) \geq 5$ if C has length 6. The first three lines of the table actually follow from inequality (2). In fact, this earlier result gives equality in these cases.

Another set of bounds arises from bounds on $\ell(a, b)$. Table 2 arises from Theorems 5.3.7, 5.4.27, 5.4.28, and 5.4.29 in [1].

3. Corollaries. It is known from Klapper [5] that for all $n \geq 3$

$$R_2(\mathbf{F}^n) = R_3(\mathbf{F}^n) = \left\lceil \frac{1}{2}n \right\rceil$$

and

$$R_4(\mathbf{F}^n) = R_5(\mathbf{F}^n) = \left\lceil \frac{1}{2}(n+1) \right\rceil.$$

TABLE 2
 Lower bounds on $R_{2^a}(C)$ for large a .

$n \geq$	$m \geq$	a	$R_{2^a}(C) \geq$
$2^{2m+1} - 2^m - 1$	1	$n - 4m$	$n - 2$
$2^{2m+1} + 2^{2m} - 2^m - 2$	2	$n - 4m - 1$	$n - 2$
$2^{2m+2} - 2^m - 2$	2	$n - 4m - 2$	$n - 2$
$2^{2m+2} + 2^{2m+1} - 2^m - 2$	2	$n - 4m - 3$	$n - 2$
$27 \cdot 2^{m-4} - 1$	4	$n - 2m$	$n - 2$
$5 \cdot 2^{m-1} - 1$	1	$n - 2m - 1$	$n - 2$
$155 \cdot 2^{m-6} - 2$	6	$n - 3m$	$n - 3$
$152 \cdot 2^{m-6} - 1$	9	$n - 3m$	$n - 3$
$3 \cdot 2^m - 1$	7	$n - 3m - 1$	$n - 3$
$1024 \cdot 2^{m-8} - 1$	4	$n - 3m - 2$	$n - 3$
$822 \cdot 2^{m-8} - 2$	8	$n - 3m - 2$	$n - 3$
$821 \cdot 2^{m-8} - 1$	13	$n - 3m - 2$	$n - 3$
$47 \cdot 2^{m-4} - 1$	11	$n - 4m$	$n - 4$
$896 \cdot 2^{m-8} - 2$	8	$n - 4m - 1$	$n - 4$
$896 \cdot 2^{m-8} - 3$	10	$n - 4m - 1$	$n - 4$
$895 \cdot 2^{m-8} - 1$	15	$n - 4m - 1$	$n - 4$
$992 \cdot 2^{m-8} - 2$	8	$n - 4m - 2$	$n - 4$
$992 \cdot 2^{m-8} - 3$	10	$n - 4m - 2$	$n - 4$
$991 \cdot 2^{m-8} - 1$	15	$n - 4m - 2$	$n - 4$
$1248 \cdot 2^{m-8} - 3$	10	$n - 4m - 3$	$n - 4$
$1247 \cdot 2^{m-8} - 1$	15	$n - 4m - 3$	$n - 4$

Using Corollary 2.2 and the known results about $K(n, R)$, the minimum cardinality of a binary code of length n and covering radius R , we can determine $R_6(\mathbf{F}^n)$ and $R_7(\mathbf{F}^n)$.

THEOREM 3.1. *For all $n \geq 4$ we have*

$$R_6(\mathbf{F}^n) = \left\lceil \frac{1}{2}(n + 1) \right\rceil$$

and

$$R_7(\mathbf{F}^n) = \left\lceil \frac{1}{2}(n + 2) \right\rceil.$$

Proof. We know—see Cohen, Lobstein, and Sloane [2] and Honkala [3]—that $K(2R+2, R) = 4$ for all $R \geq 1$, $K(2R+3, R) = 7$ for all $R \geq 1$, and $K(2R+4, R) \geq 8$ for all $R \geq 0$.

By Lemma 1.1 this implies that $t_1(n, 6) = \frac{1}{2}(n - 1)$ for odd $n \geq 5$ and $t_1(n, 6) = \frac{1}{2}(n - 2)$ for even $n \geq 4$. Hence $t_1(n, 6) = \lfloor \frac{1}{2}(n - 1) \rfloor$ and, by Corollary 2.2,

$$R_6(\mathbf{F}^n) = n - t_1(n, 6) = \left\lceil \frac{1}{2}(n + 1) \right\rceil.$$

Similarly, $t_1(n, 7) = \frac{1}{2}(n - 3)$ for all odd $n \geq 5$ and $t_1(n, 7) = \frac{1}{2}(n - 2)$ for even $n \geq 4$. Hence $t_1(n, 7) = \lfloor \frac{1}{2}(n - 2) \rfloor$ and, by Corollary 2.2,

$$R_7(\mathbf{F}^n) = n - t_1(n, 7) = \left\lceil \frac{1}{2}(n + 2) \right\rceil. \quad \square$$

Using Corollary 2.2 and the results in section 12.5 of Cohen et al. [1] we obtain asymptotic results on $R_m(\mathbf{F}^n)$. For instance, using Theorems 12.5.1 (sphere-covering bound) and 12.5.10 (from Lovász, Spencer, and Vesztergombi [7]) we obtain the following two theorems.

THEOREM 3.2. *For all n and m ,*

$$R_m(\mathbf{F}^n) \leq \frac{1}{2}n + \sqrt{n \log_2 m \ln 2/2}.$$

THEOREM 3.3. *For all n and m ,*

$$R_m(\mathbf{F}^n) \leq \frac{1}{2}n + 12\sqrt{m}.$$

4. On the 3-covering radius of Hamming codes. Let \mathcal{H}_r denote the Hamming code of order r . It was shown by Klapper [5] that for any $m \geq 2$ and $r \geq 2$,

$$2^{r-1} \leq R_m(\mathcal{H}_r) \leq 2^{r-1} + c_m,$$

where c_m is a constant depending only on m . It was also shown that $R_m(\mathcal{H}_2) = 3$ for $m \geq 2$; for $r \geq 3$ we have $R_2(\mathcal{H}_r) = 2^{r-1}$; and for $m = 3, 4, 5$ we have

$$2^{r-1} \leq R_m(\mathcal{H}_r) \leq 2^{r-1} + 1.$$

However, in this last case the precise value was unknown. In this section, using Theorem 2.1, we determine exactly the 3-covering radius of the Hamming codes. The proof is based on the following lemma.

LEMMA 4.1. *A binary code of odd length n , cardinality three, and covering radius $\frac{1}{2}(n - 1)$ contains a word-complement pair.*

Proof.

Step 1. We first show that the covering radius of the code consisting of the three codewords

$$\begin{array}{lll} c_1 & 11 \dots 1 & 00 \dots 0 & 00 \dots 0 \\ c_2 & 00 \dots 0 & 11 \dots 1 & 00 \dots 0 \\ c_3 & \underbrace{00 \dots 0}_{\alpha} & \underbrace{00 \dots 0}_{\beta} & \underbrace{11 \dots 1}_{\gamma}, \end{array}$$

where $\alpha \leq \beta \leq \gamma$ equals

$$t = \alpha + \left\lfloor \frac{\beta + \gamma}{2} \right\rfloor.$$

For every $x \in \mathbf{F}^n$ we have $d(x, C) \leq \lfloor \frac{1}{2}(d(x, c_2) + d(x, c_3)) \rfloor \leq t$. On the other hand, take $x \in \mathbf{F}^n$ which has α ones in the beginning, then $\lceil \frac{1}{2}(\alpha + \beta) \rceil$ ones among the next β and $\lceil \frac{1}{2}(\alpha + \gamma) \rceil$ ones among the last γ coordinates. Then

$$d(x, c_1) = \left\lceil \frac{1}{2}(\alpha + \beta) \right\rceil + \left\lceil \frac{1}{2}(\alpha + \gamma) \right\rceil,$$

$$d(x, c_2) = \alpha + \left(\beta - \left\lceil \frac{1}{2}(\alpha + \beta) \right\rceil \right) + \left\lceil \frac{1}{2}(\alpha + \gamma) \right\rceil = \left\lfloor \frac{1}{2}(\alpha + \beta) \right\rfloor + \left\lceil \frac{1}{2}(\alpha + \gamma) \right\rceil,$$

and

$$d(x, c_3) = \alpha + \left\lceil \frac{1}{2}(\alpha + \beta) \right\rceil + \left(\gamma - \left\lceil \frac{1}{2}(\alpha + \gamma) \right\rceil \right) = \left\lceil \frac{1}{2}(\alpha + \beta) \right\rceil + \left\lfloor \frac{1}{2}(\alpha + \gamma) \right\rfloor.$$

Because $d(x, c_1) \geq d(x, c_2)$, it suffices to show that $d(x, c_2) \geq t$ and $d(x, c_3) \geq t$. If β and γ have the same parity, then $d(x, c_2)$ and $d(x, c_3)$ both equal t . If β and γ have different parities, then $t = \alpha + \frac{1}{2}(\beta + \gamma - 1)$, and exactly one of $d(x, c_2)$ and $d(x, c_3)$ equals t and the other $t + 1$. Hence $d(x, C) = t$, proving that C has covering radius t .

Step 2. Now assume that we have a code of odd length n with three codewords and covering radius $\frac{1}{2}(n - 1)$. By taking a suitable translate, if necessary, we may assume that in each coordinate all codewords have 0's or at most one of the codewords has 1. Assume that the number of identically zero coordinates is i and that by puncturing these i coordinates we obtain the code C in Step 1 of length $n - i$. By Step 1, the covering radius of our original code equals

$$s = i + \alpha + \left\lfloor \frac{1}{2}(\beta + \gamma) \right\rfloor,$$

and in particular

$$s \geq i + \left\lfloor \frac{1}{2}(n - i) \right\rfloor > \frac{1}{2}(n - 1)$$

if $i > 0$. Hence $i = 0$ and

$$s = \left\lfloor \frac{1}{2}(n + \alpha) \right\rfloor > \frac{1}{2}(n - 1)$$

unless $\alpha = 0$. Hence $\alpha = 0$, and c_2 is the complement of c_3 . \square

THEOREM 4.2. $t_1(3, \mathcal{H}_r) = \frac{1}{2}(n - 1) = 2^{r-1} - 1$ for all $r \geq 3$.

Proof. Assume that C consists of three codewords c_1, c_2 , and c_3 of length $n = 2^r - 1$ such that the balls of radius $\frac{1}{2}(n - 3) = 2^{r-1} - 2$ centered at the codewords of C contain all the codewords of the Hamming code \mathcal{H}_r . Because the covering radius of the Hamming code is one, this implies that the balls of radius $2^{r-1} - 1$ centered at the words c_1, c_2 , and c_3 cover the whole space \mathbf{F}^n . By the previous lemma this is only possible if the set $\{c_1, c_2, c_3\}$ contains a word-complement pair: say, c_2 is the

complement of c_3 . However, we know that $R_2(\mathcal{H}_r) = \frac{1}{2}(n+1) = 2^{r-1}$ for $r \geq 3$ and that there is a codeword $c \in \mathcal{H}_r$ such that $d(c, c_2), d(c, c_3) \in \{\frac{1}{2}(n-1), \frac{1}{2}(n+1)\}$. The Hamming code is self-complementary: it is linear and the all-one vector is a codeword because the sum of all columns in its parity check matrix is the zero column. Therefore also $\bar{c} \in \mathcal{H}_r$. Neither c nor \bar{c} is contained in the spheres $B_{(n-3)/2}(c_2)$ and $B_{(n-3)/2}(c_3)$. Since their mutual distance is n , they cannot both belong to $B_{(n-3)/2}(c_1)$ either. This contradiction proves that $t_1(3, \mathcal{H}_r) \geq \frac{1}{2}(n-1)$. The opposite inequality is clear. \square

THEOREM 4.3. $R_3(\mathcal{H}_r) = \frac{1}{2}(n+1) = 2^{r-1}$ for all $r \geq 3$.

Proof. This now immediately follows from Theorems 2.1 and 4.2. \square

5. A sphere bound.

THEOREM 5.1. *Suppose C is a code with length n , and $m < |C|$. Then*

$$R_m(C) \leq n - \frac{1}{2}d_0,$$

where d_0 is the largest minimum distance among the $(m+1)$ -element subcodes of C . In particular, if the minimum distance of C is d , then $R_m(C) \leq n - \frac{1}{2}d$.

Proof. By Theorem 2.1 it suffices to prove that $t_1(m, C) \geq \frac{1}{2}d_0$. If not, $t_1(m, C) < \frac{1}{2}d_0$, which is impossible because we know that C has an $(m+1)$ -element subcode C_0 with minimum distance d_0 and therefore no ball $B_t(x)$ with $t < \frac{1}{2}d_0$ can cover more than one element of C_0 . \square

Acknowledgment. The authors would like to thank an anonymous referee for useful comments.

REFERENCES

- [1] G. COHEN, I. HONKALA, S. LITSYN, AND A. LOBSTEIN, *Covering Codes*, Elsevier, Amsterdam, 1997.
- [2] G. D. COHEN, A. C. LOBSTEIN, AND N. J. A. SLOANE, *Further results on the covering radius of codes*, IEEE Trans. Inform. Theory, 32 (1986), pp. 680–694.
- [3] I. S. HONKALA, *Modified bounds for covering codes*, IEEE Trans. Inform. Theory, 37 (1991), pp. 351–365.
- [4] I. HONKALA AND A. KLAPPER, *Bounds for the multicovering radii of Reed-Muller codes with applications to stream ciphers*, Des. Codes Cryptogr., 23 (2001), pp. 131–145.
- [5] A. KLAPPER, *The Multicovering radii of codes*, IEEE Trans. Inform. Theory, 43 (1997), pp. 1372–1377.
- [6] A. KLAPPER, *Improved lower bounds for multicovering codes*, IEEE Trans. Inform. Theory, 45 (1999), pp. 2532–2534.
- [7] L. LOVÁSZ, J. H. SPENCER, AND K. VESZTERGOMBI, *Discrepancy of set-systems and matrices*, European J. Combin., 7 (1986), pp. 151–160.

PARTIAL CUBES AND CROSSING GRAPHS*

SANDI KLAVŽAR[†] AND HENRY MARTYN MULDER[‡]

Abstract. Partial cubes are defined as isometric subgraphs of hypercubes. For a partial cube G , its crossing graph $G^\#$ is introduced as the graph whose vertices are the equivalence classes of the Djoković–Winkler relation Θ , two vertices being adjacent if they cross on a common cycle. It is shown that every graph is the crossing graph of some median graph and that a partial cube G is 2-connected if and only if $G^\#$ is connected. A partial cube G has a triangle-free crossing graph if and only if G is a cube-free median graph. This result is used to characterize the partial cubes having a tree or a forest as its crossing graph. An expansion theorem is given for the partial cubes with complete crossing graphs. Cartesian products are also considered. In particular, it is proved that $G^\#$ is a complete bipartite graph if and only if G is the Cartesian product of two trees.

Key words. isometric subgraph, hypercube, partial cube, crossing graph, median graph, triangle-free graph, Cartesian product of graphs

AMS subject classifications. 05C75, 05C12

PII. S0895480101383202

1. Introduction. A *partial cube* is a connected graph that admits an isometric embedding into a hypercube. Partial cubes have first been investigated in the 1970s by Graham and Pollak [13], who used them as a model for a communication network. By now, the structure of partial cubes is relatively well understood. Djoković [10] characterized these graphs via convexity of certain vertex partitions. He also introduced the relation Θ on the edge set of a graph. This relation was later used by Winkler [28] to characterize the partial cubes as those bipartite graphs for which Θ is transitive. Chepoi [8] followed with an expansion theorem for partial cubes. Another characterization of partial cubes was obtained by Avis [4]; cf. also [27]. Partial cubes have found several applications. See, for instance, [11] for connections with oriented matroids and [9, 21] for recent applications to chemical graph theory. An important subclass of the class of partial cubes is that of the median graphs (see [24, 25]); cf. [22]. Among the median graphs the cube-free median graphs stand out; see, for instance, [5, 20, 23].

The fastest known recognition algorithm for partial cubes is of complexity $O(mn)$, where n and m are the number of vertices and edges of a given graph. Since for partial cubes $m \leq (n \log n)/2$ (cf. [2, 3, 12, 19]), this complexity reduces to $O(n^2 \log n)$. The first such algorithm is due to Aurenhammer and Hagauer [2, 3]. Another more general algorithm for recognizing partial Hamming graphs (isometric subgraphs of Cartesian products of complete graphs) of complexity $O(mn)$ is given in [1]. Applying the canonical isometric embedding theory of Graham and Winkler [14], a simple algorithm for recognizing partial Hamming graphs of the same complexity can be obtained; see [17, 19]. However, only a trivial lower bound $O(m)$ for recognizing partial cubes

*Received by the editors January 5, 2001; accepted for publication (in revised form) February 8, 2002; published electronically April 3, 2002.

<http://www.siam.org/journals/sidma/15-2/38320.html>

[†]Department of Mathematics, PEF, University of Maribor, Koroška cesta 160, 2000 Maribor, Slovenia (sandi.klavzar@uni-lj.si). This author's research was supported by the Ministry of Education, Science, and Sport of Slovenia under grant 0101-P-504.

[‡]Econometrisch Instituut, Erasmus Universiteit, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands (hmmulder@few.eur.nl). This author's research was supported by the Ministry of Education, Science, and Sport of Slovenia and the University of Maribor, Slovenia.

is known. This contrasts with the recognition problem for median graphs, where the connection between median graphs and triangle-free graphs [20] provides strong evidence that the fastest known recognition algorithms for median graphs [15, 19] are close to being optimal.

In this paper we are interested in the structure of the Θ -classes of a partial cube G . An important feature is whether two Θ -classes cross or not. We say that two Θ -classes F_1 and F_2 cross in G if edges of F_2 occur in both the components of $G - F_1$. The *crossing graph* $G^\#$ of a partial cube G has the Θ -classes of G as its vertices, where two vertices of $G^\#$ are joined by an edge whenever they cross as Θ -classes in G .

In the next section we recall concepts needed later and collect basic properties of the relation Θ . Our results are presented in sections 3–6. We start with a theorem asserting that every connected graph is the crossing graph of some partial cube, even the crossing graph of some median graph. Thus at first sight the notion of a crossing graph may not seem very interesting. However, appearances are deceptive. There is a nontrivial relationship between the structure of a partial cube and that of its crossing graph. For instance, we prove the following results for partial cubes G : “ G is 2-connected if and only if its crossing graph is connected,” “the crossing graph of G is triangle-free if and only if G is a cube-free median graph,” “the crossing graph of G is a tree if and only if G is a 2-connected cube-free median graph with some forbidden subgraphs,” and “the crossing graph of G is a complete bipartite graph if and only if G is the Cartesian product of two trees.” Along the way some other types of graphs, such as C_4 -trees and C_4 -cactoids, are considered. Moreover, we characterize the partial cubes with a complete graph as crossing graph. We conclude this paper with a number of open problems.

2. Preliminaries. For $u, v \in V(G)$, let $d_G(u, v)$ denote the length of a shortest path (also called *geodesic*) in G from u to v . A subgraph H of a graph G is an *isometric* subgraph if $d_H(u, v) = d_G(u, v)$ for all $u, v \in V(H)$. The *interval* $I(u, v)$ between two vertices u and v in G is the set of all vertices on shortest paths between u and v . A subgraph H of G is *convex* if we have $I(u, v) \subseteq V(H)$ for any $u, v \in V(H)$.

The *Cartesian product* $G \square H$ of two graphs G and H is the graph with vertex set $V(G) \times V(H)$ and $(a, x)(b, y) \in E(G \square H)$ whenever either $ab \in E(G)$ and $x = y$ or $a = b$ and $xy \in E(H)$. The n -cube Q_n is the Cartesian product of n copies of the complete graph on two vertices K_2 .

For a graph $G = (V, E)$ and $X \subseteq V$, let $\langle X \rangle$ denote the subgraph induced by X . For two vertices u and v on a path P , we denote the subpath of P between u and v by $u \rightarrow \dots P \dots \rightarrow v$.

The Djoković–Winkler relation Θ [10, 28] is defined on the edge set of a graph in the following way. Edges $e = xy$ and $f = uv$ of a graph G are in relation Θ if

$$d_G(x, u) + d_G(y, v) \neq d_G(x, v) + d_G(y, u).$$

Relation Θ is reflexive and symmetric. If G is bipartite, then Θ can be defined as follows: $e = xy$ and $f = uv$ are in relation Θ if

$$d(x, u) = d(y, v) \quad \text{and} \quad d(x, v) = d(y, u).$$

Among bipartite graphs, Θ is transitive precisely for partial cubes (i.e., isometric subgraphs of hypercubes), as has been proved by Winkler in [28].

Let $G = (V, E)$ be a connected, bipartite graph. For any edge ab of G we write

$$\begin{aligned} W_{ab} &= \{w \in V \mid d_G(a, w) < d_G(b, w)\}, \\ U_{ab} &= \{w \in W_{ab} \mid w \text{ has a neighbor in } W_{ba}\}, \\ F_{ab} &= \{e \in E \mid e \text{ is an edge between } W_{ab} \text{ and } W_{ba}\}, \\ G_{ab} &= \langle W_{ab} \rangle. \end{aligned}$$

Note that if G is bipartite, then we have $V = W_{ab} \cup W_{ba}$. For a bipartite graph G , the sets F_{ab} are called *colors* and the subgraphs G_{ab}, G_{ba} form the *split* of the color F_{ab} . The subgraph $\langle U_{ab} \rangle$ is the *side* of color F_{ab} in G_{ab} , and $\langle U_{ba} \rangle$ is the *opposite* of $\langle U_{ab} \rangle$. Djoković [10] characterized the partial cubes as the connected bipartite graphs in which all subgraphs G_{ab} are convex.

We now state three well-known facts about the relation Θ ; cf. [18].

LEMMA 2.1. *Let G be a connected, bipartite graph, and let ab be any edge of G . Then F_{ab} is the set of all edges in relation Θ with ab .*

Note that for partial cubes Lemma 2.1 asserts that Θ -classes coincide with the sets F_{ab} , a fact that will be used implicitly in what follows.

We say that a color *occurs* in a subgraph H if there is an edge of that color in H .

LEMMA 2.2. *Let C be an isometric cycle of a partial cube G , and let F_{ab} be a color which occurs in C . Then F_{ab} occurs in C exactly twice (in two antipodal edges).*

LEMMA 2.3. *Suppose P is a path connecting the endpoints of an edge e . Then P contains an edge f with $e\Theta f$.*

The conclusion of Lemma 2.3 holds also if P is a walk, as every walk containing the endpoints of e contains a path between the endpoints of e ; cf. Lemma 2.4 of [19].

For our purposes it is convenient to have statements available that are slightly stronger than the above lemmas. These may also be part of folklore, but to make the paper self-contained we provide them with proofs.

LEMMA 2.4. *Let G be a partial cube. Then a path P in G is a geodesic if and only if no color occurs twice on P .*

Proof. If P is a geodesic, then, by the definition of Θ , all colors on P must be distinct.

Conversely, let $P = v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_n$ be a path on which all colors are distinct. Assume that P is not a geodesic with n as small as possible. Note that $n \geq 3$. Then we have $d(v_0, v_{n-1}) = d(v_1, v_n) = n - 1$, $d(v_0, v_n) = n - 2$. Let Q be a v_0, v_n -geodesic. By minimality, P and Q are internally disjoint. By Lemma 2.3, the edge v_0v_1 is in relation Θ with an edge of the cycle composed of P and Q . Moreover, $v_1 \rightarrow v_0 \rightarrow \dots \rightarrow Q \rightarrow v_n$ is a path of length $n - 1$ and thus a geodesic; hence v_0v_1 is not in relation Θ with any edge of Q . So P contains two edges of the same color, a contradiction. \square

Let C be an even cycle of length $2k$. We call two edges on C *antipodal* if their endpoints are joined by two paths of length $k - 1$ on C .

LEMMA 2.5. *Let G be a partial cube. Then C is an isometric cycle in G if and only if every color on C occurs only on antipodal edges.*

Proof. Let C be an isometric cycle of length $2k$ in G . Then every path of length at most k on C is a geodesic in G , so that, by Lemma 2.4, all colors on such paths occur exactly once. Since each color on C occurs at least twice, it follows that each color on C occurs precisely on antipodal edges.

Conversely, let C be a cycle of length $2k$ in G such that each color on C occurs precisely on antipodal edges. Then, by Lemma 2.4, each path of length at most k must be a geodesic in G . Hence C is isometric in G . \square

LEMMA 2.6. *Let G be a partial cube, and let F_{ab} be a color of G . If uv, xy are edges of F_{ab} , with u, x on one side and v, y on the opposite, and if P is any u, x -*

geodesic and Q is any v, y -geodesic, then P and Q contain the same colors, and each color occurs at most once on P and at most once on Q .

Proof. By definition of a color, a geodesic contains every color at most once. Take any color on P , say, edge e is of that color. Then $u \rightarrow \cdots P \cdots \rightarrow x \rightarrow y \rightarrow \cdots Q \cdots \rightarrow v \rightarrow u$ is a cycle through e , whence constitutes a path between the ends of e . Hence, by Lemma 2.3, it contains an edge f of the same color. Since f cannot be on P , it is on Q . Conversely, every color on Q occurs on P . \square

A *median graph* is, by definition, a connected graph such that, for every triple of its vertices, there is a unique vertex lying on a geodesic (i.e., shortest path) between each pair of the triple. It follows immediately from the definition that median graphs are bipartite. By now, the class of median graphs has been well investigated and a rich structure theory is available; see the recent survey [22]. Median graphs are partial cubes (see [24, 25]), whence relation Θ is transitive on median graphs. They may be characterized as the connected bipartite graphs in which all subgraphs $\langle U_{ab} \rangle$ are convex; cf. [6]. Another relevant feature of median graphs is that any isometric cycle of length $2n$ is contained in an induced Q_n . (This again can be deduced directly from the definition.)

3. Crossing graphs. Let G be a partial cube. We say that two colors *cross* if their splits G_1, G_2 and H_1, H_2 satisfy $G_i \cap H_j \neq \emptyset$ for $1 \leq i, j \leq 2$; see [23]. The *crossing graph* $G^\#$ of a partial cube G has the colors of G as its vertices, and two vertices are adjacent if they cross as colors.

At first sight it is not clear which graphs are crossing graphs. However, using the following concept, the answer is clear. For a graph G , the *simplex graph* $S(G)$ of G is the graph whose vertices are the complete subgraphs of G (including the empty graph), two vertices being adjacent if, as complete subgraphs of G , they differ in exactly one vertex; see [7]. It is easily seen that a simplex graph is a median graph, and hence a partial cube, by checking that it satisfies the definition of a median graph.

THEOREM 3.1. *Every graph is a crossing graph of some median graph. More precisely, for any graph G we have $G = S(G)^\#$.*

Proof. Let $V(G) = \{1, 2, \dots, n\}$. Since vertex \emptyset of $S(G)$ is of degree n , we infer that $S(G)^\#$ has at least n vertices. Let uv be an arbitrary edge of $S(G)$. Without loss of generality we may assume that $u = \{1, 2, \dots, k\}$ and $v = \{1, 2, \dots, k+1\}$. It is now straightforward to check that the edge $(\emptyset, \{k+1\})$ is in relation Θ with uv . It follows that $S(G)^\#$ has exactly n vertices and that vertex i of G corresponds to the color of edge $(\emptyset, \{i\})$ in $S(G)$. Assume that vertices 1 and 2 are adjacent in G . Then $\emptyset, \{1\}, \{2\}$, and $\{1, 2\}$ induce C_4 in $S(G)$, and so the corresponding colors cross. Finally, if 1 is not adjacent to 2, then they are not in the same complete subgraph of G , which implies that the corresponding colors do not cross in $S(G)$. \square

There is another (simplified) construction showing that every graph is the crossing graph of some partial cube. For a graph G , let \tilde{G} be the graph obtained from G by subdividing all edges of G and adding a new vertex z joined to all the original vertices of G ; see [20]. Then we can argue similarly as above that for any graph G we have $G = \tilde{G}^\#$.

To prove relations between properties of a partial cube and properties of its crossing graph, we need some simple criteria for colors to determine whether they cross.

LEMMA 3.2. *Let G be a partial cube. Then any cycle of G contains two crossing colors.*

Proof. Let C be any cycle of G . Note that each color occurs an even number of times on C . Choose two edges uv and xy of the same color F on C such that on the

subpath $P = u \rightarrow v \rightarrow \dots \rightarrow x \rightarrow y$ of C every color occurs at most once between u and x . Since uv and xy cannot be adjacent, there is at least one other color on P . By Lemma 2.3, color F crosses with each color on the subpath $v \rightarrow \dots \rightarrow x$ of P . \square

We say that two colors *alternate on a cycle C* if they both occur in C and we encounter them alternately while walking along C . Note that Lemma 2.5 in particular implies that any two colors on an isometric cycle of a partial cube alternate.

LEMMA 3.3. *Let G be a partial cube G , and let F_{ab} and F_{uv} be two different colors of G . Then the following statements are equivalent:*

- (i) F_{ab} and F_{uv} cross.
- (ii) F_{ab} and F_{uv} alternate on an isometric cycle of G .
- (iii) F_{ab} and F_{uv} occur on an isometric cycle of G .
- (iv) Each of the colors F_{ab} and F_{uv} appear exactly twice on a cycle of G and they alternate.

Proof. (i) \Rightarrow (ii) Suppose that the colors F_{ab} and F_{uv} cross. We may assume that uv lies in G_{ab} . As the colors cross, there is an edge $u'v'$ in G_{ba} of color F_{uv} with u and u' on the one side of F_{uv} and v and v' on the opposite, that is, $d_G(u, u') = d_G(v, v') = d_G(u, v') - 1$. We choose the edges uv and $u'v'$ such that $d_G(u, u')$ is as small as possible. Let $l(S)$ denote the length of the walk S in G .

Let P be a shortest u, u' -path, and let Q be a shortest v, v' -path, so that P lies in G_{uv} and Q lies in G_{vu} . The paths P and Q are disjoint and each of them contains exactly one edge of F_{ab} .

We claim that $C = u \rightarrow \dots P \dots \rightarrow u' \rightarrow v' \rightarrow \dots Q \dots \rightarrow v \rightarrow u$ is an isometric cycle. Assume the contrary, and let x and y be vertices of C such that $d_G(x, y) < d_C(x, y)$. Let R be a shortest x, y -path. We may select x and y such that R is internally disjoint from C and that x lies on P and y on Q . Then $C' = u \rightarrow \dots P \dots \rightarrow x \rightarrow \dots R \dots \rightarrow y \rightarrow \dots Q \dots \rightarrow v \rightarrow u$ is a cycle of length $l(C') < l(C)$. By Lemma 2.3, there is an edge $x'y'$ of color F_{uv} on C' with u, x' , and u' on the one side and v, y' , and v' on the opposite of F_{uv} . Write $P' = u \rightarrow \dots P \dots \rightarrow x \rightarrow \dots R \dots \rightarrow x'$ and $Q' = v \rightarrow \dots Q \dots \rightarrow y \rightarrow \dots R \dots \rightarrow y'$. Then we have

$$2d_G(u, u') = d_G(u, u') + d_G(v, v') = l(C) - 2$$

$$> l(C') - 2 = l(P') + l(Q') \geq d_G(u, x') + d_G(v, y') = 2d_G(u, x').$$

Hence we have $d_G(u, x') < d_G(u, u')$, which contradicts the minimality of $d_G(u, u')$. Thus we conclude that C is an isometric cycle. By Lemma 2.5 each of the two colors appears exactly twice on C and the colors alternate on C .

(ii) \Rightarrow (iii) This implication is trivial.

(iii) \Rightarrow (iv) This follows from Lemma 2.5.

(iv) \Rightarrow (i) Let C be a cycle of G on which the colors F_{ab} and F_{uv} appear exactly twice. Let $a'b'$ and $u'v'$ be the second edges of colors F_{ab} and F_{uv} , respectively, and let $d_G(a, a') = d_G(b, b') = d_G(a, b') - 1$. Then the cycle C can be written as $C = a \rightarrow \dots P \dots \rightarrow a' \rightarrow b' \rightarrow \dots \rightarrow Q \dots \rightarrow b \rightarrow a$, where P and Q are the corresponding paths on C connecting a with a' and b' with b . Since ab and $a'b'$ are the only edges from F_{ab} on C , we observe that P lies in G_{ab} and Q lies in G_{ba} . Moreover, as the colors alternate on C , we may assume that uv lies on P and $u'v'$ lies on Q . Without loss of generality we may assume that $a \in W_{uv}$ and $b \in W_{u'v'}$. Then we have $a \in W_{ab} \cap W_{uv}$, $a' \in W_{ab} \cap W_{vu}$, $b \in W_{ba} \cap W_{uv}$ and $b' \in W_{ba} \cap W_{vu}$. Thus the colors F_{ab} and F_{uv} cross. \square

Our first result that relates properties of the crossing graph $G^\#$ to properties of the partial cube G involves connectivity.

THEOREM 3.4. *Let $G = (V, E)$ be a partial cube. Then G is 2-connected if and only if $G^\#$ is connected.*

Proof. First assume that G is not 2-connected, and let x be a cutvertex in G . Let A be a subgraph of G induced by x and one component of $G - x$, and let B be the subgraph of G induced by x and the remaining part of $G - x$. Then A and B both contain edges, and we have $A \cup B = G$ and $A \cap B = \{x\}$, so that no color in A crosses with a color in B . Hence, in $G^\#$, there is no path between any color in A and any color in B ; that is, $G^\#$ is disconnected.

Conversely, let G be 2-connected and take any two incident edges uv and vw of G , with, say, uv colored red and vw colored blue. Since G is 2-connected, there exists a path between u and w in G not containing v . Let P be such a u, w -path of minimal length k . Since $P \rightarrow v \rightarrow u$ is a cycle, red and blue must occur on P . Now we walk along P from u to w . Let xy be the first red or blue edge on P , where we traverse xy from x to y .

Suppose the color of xy is blue, that is, the color of vw . Then x and v are on the same side of color blue and y and w are on the opposite. Let Q be a geodesic between x and v , and let Q' be a geodesic between y and w . Then by Lemma 2.3 red occurs on Q and, since Q is a geodesic, red occurs only once on Q .

By Lemma 2.6 red occurs exactly once also on Q' ; so red and blue occur exactly twice alternately on the cycle

$$v \rightarrow \cdots Q \cdots \rightarrow x \rightarrow y \rightarrow \cdots Q' \cdots \rightarrow w \rightarrow v.$$

So, by Lemma 3.3 (iv), red and blue cross in G and are adjacent in $G^\#$.

Suppose the color of xy is red, that is, the color of uv . Now u and x are on one side of red, and v and y are on the opposite. Let P_1 be a geodesic between v and y , and let u_1 be the neighbor of v on P_1 . Note that the u, x -subpath of P is a geodesic by minimality of P . Thus, using Lemma 2.3, red and the color of vu_1 cross in G , and so are adjacent in $G^\#$. Now,

$$u_1 \rightarrow \cdots P_1 \cdots \rightarrow y \rightarrow \cdots P \cdots \rightarrow w$$

is a walk between u_1 and w not containing v of length $k - 2$. Hence there is a path between u_1 and w not containing v of length at most $k - 2$.

Repeating the above argument, we find neighbors u_2, \dots, u_p of v such that the colors of vu_i and vu_{i+1} cross, for $i = 1, \dots, p - 1$, and also the color of vu_p and blue cross. Thus we have constructed a path in $G^\#$ between red and blue. Connectivity of $G^\#$ now follows from the connectivity of G . \square

Note that $P_n \square P_n$ is 2-connected but not 3-connected, since it contains a vertex of degree 2. On the other hand, $(P_n \square P_n)^\# = K_{n,n}$ is n -connected. So there does not exist an analogue of Theorem 3.4 for higher connectivities.

4. Complete crossing graphs. In this section we consider the partial cubes that have complete crossing graphs. In [23] it was proved that in a median graph G there are n pairwise crossing colors if and only if G contains an induced n -cube. We restate this result in the next proposition and its corollary.

PROPOSITION 4.1. *A median graph is a hypercube if and only if its crossing graph is complete. More precisely, if G is a median graph, then $G = Q_n$, $n \geq 1$ if and only if $G^\# = K_n$.*

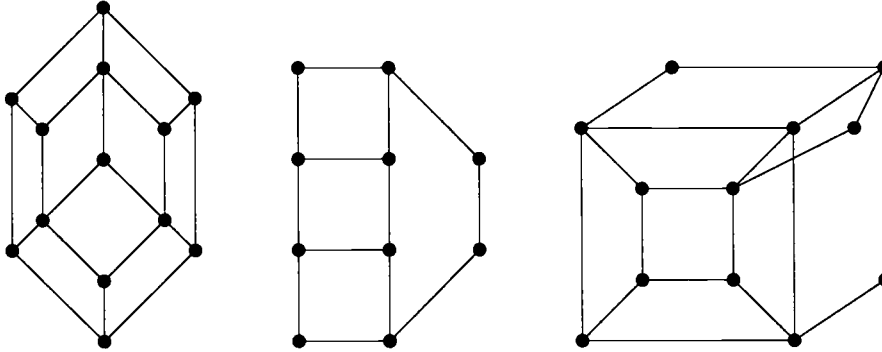


FIG. 4.1. *Partial cubes with complete crossing graphs.*

Recall that the clique number of a graph is the size of a largest complete subgraph in the graph.

COROLLARY 4.2. *Let G be a median graph. Then the clique number of $G^\#$ is equal to the dimension of the largest hypercube in G .*

A simple consequence of the above cited result is the following corollary; see [23].

COROLLARY 4.3. *Let G be a partial cube. Then G is a tree if and only if $G^\#$ is the complement of a complete graph.*

For partial cubes the variety of graphs with complete crossing graphs is much richer than for median graphs. Besides hypercubes one finds even cycles, Q_3 minus a vertex, and the graphs from Figure 4.1. Moreover, the Cartesian product preserves this property (cf. Proposition 6.1).

In order to characterize the partial cubes with complete crossing graphs, we recall the concept of expansion; see [24, 25, 26] or [19].

Let G' be a connected graph. A *proper cover* G'_1, G'_2 consists of two induced subgraphs G'_1, G'_2 of G' such that $G' = G'_1 \cup G'_2$ and $G'_0 = G'_1 \cap G'_2$ is a nonempty subgraph, called the *intersection* of the cover. The cover is *isometric* (resp., *convex*) if it consists of isometric (resp., convex) subgraphs.

Let G' be a connected graph, and let G'_1, G'_2 be a proper cover of G' with $G'_0 = G'_1 \cap G'_2$. The *expansion* of G' with respect to G'_1, G'_2 is the graph G constructed as follows. Let G_i be an isomorphic copy of G'_i , for $i = 1, 2$, and, for any vertex u' in G'_0 , let u_i be the corresponding vertex in G_i for $i = 1, 2$. Then G is obtained from the disjoint union $G_1 \cup G_2$, where for each u' in G'_0 the vertices u_1 and u_2 are joined by an edge. We denote the copy of G'_0 in G_i by G_{0i} for $i = 1, 2$. Note that the set F of edges between G_{01} and G_{02} is a Θ -class, i.e., a color, with sides G_{01} and G_{02} . If the cover G'_1, G'_2 is isometric (resp., convex), then we call G an *isometric* (resp., *convex*) expansion. Finally, G is an *all-color expansion* if any of the G'_1 and G'_2 contains at least one edge of each Θ -class of G . The converse operation of expansion is *contraction*: G' is the contraction of G with respect to the split G_1, G_2 , or, equivalently, with respect to the color F .

Chepoi [8] proved that a graph G is a partial cube if and only if G is obtained from the one-vertex graph K_1 by successive isometric expansions. Mulder [24, 25] proved that G is a median graph if and only if G can be obtained from K_1 by successive

convex expansions.

Let G be a partial cube with a complete crossing graph, and let H be an isometric subgraph of G that meets all the Θ -classes of G . Then the expansion of G with respect to H and G is an all-color expansion, and the expanded graph has a complete crossing graph. (Note that the right-hand graph of Figure 4.1 is an expansion of Q_3 with respect to Q_3 and $K_{1,3}$.) More generally, we have the following result.

PROPOSITION 4.4. *Let G be a partial cube. Then $G^\#$ is a complete graph if and only if G can be obtained from K_1 by a sequence of all-color expansions.*

Proof. Assume first that $G^\#$ is a complete graph, and let F_{ab} be an arbitrary but fixed color of G . Let G' be the contraction of G with respect to F_{ab} , and let G'_1, G'_2 be the corresponding cover of G' , so that G is the expansion of G' with respect to the cover G'_1, G'_2 . Let F_{uv} be any other color of G . Then, since $G^\#$ is complete, we infer from Lemma 3.3 (ii) (or (iv)) that both G_1 and G_2 contain an edge from F_{uv} . Hence G'_1 and G'_2 both contain edges of this color. Induction completes the argument.

Conversely, suppose that G can be obtained from K_1 by a sequence of all-color expansions. Let G be an all-color expansion of G' with respect to the cover G'_1, G'_2 , and let F_{ab} be the color of this expansion step. We need to show that F_{ab} crosses with any other color. Let F_{uv} be an arbitrary color different from F_{ab} . Since G is obtained by an all-color expansion, there is an edge xx' from F_{uv} in G_1 and an edge yy' from F_{uv} in G_2 with x and y on the one side and x' and y' on the opposite. Let P be a shortest x, y -path, and let Q be a shortest x', y' -path. Then $x' \rightarrow x \rightarrow P$ is a shortest path and thus no edge of P belongs to F_{uv} . Similarly, we see that no edge of Q is in F_{uv} . Moreover, there are exactly two edges from F_{ab} in the cycle $x' \rightarrow x \rightarrow \dots P \dots \rightarrow y \rightarrow y' \rightarrow \dots Q \dots \rightarrow x' \rightarrow x$. Therefore, by Lemma 3.3 (iv) the colors F_{ab} and F_{uv} cross. \square

5. Triangle-free crossing graphs. *Cube-free median graphs* are median graphs that do not contain Q_3 as an induced subgraph. Note that a median graph is cube-free if and only if it does not contain isometric cycles of length at least 6. Moreover, each side of any color in a cube-free median graph must be a tree.

The class of cube-free median graphs may seem a rather special class of graphs. However, in [20] it was proved that there exists a one-to-one correspondence between the class of triangle-free graphs and a special subclass of cube-free median graphs. Hence, in the universe of all graphs, the density of the triangle-free graphs is as large as that of the cube-free median graphs (being triangle-free themselves). In [23] it was shown that cube-free median graphs play a special role in the theory of consensus functions on graphs. In our next result we show that the condition that the crossing graph of a partial cube G is triangle-free turns out to be a rather strong condition—it is equivalent to the fact that G is a cube-free median graph.

THEOREM 5.1. *Let G be a partial cube. Then $G^\#$ is triangle-free if and only if G is a cube-free median graph.*

Proof. First let G be a cube-free median graph. Then, by Theorem 11 from [23], we know that G does not contain three mutually crossing colors, so that $G^\#$ is triangle-free.

Conversely, let $G^\#$ be triangle-free; that is, G does not contain three mutually crossing colors. Take any color F in G , say between G_1 and G_2 , with sides G_{01} and G_{02} , respectively. (Here and later we use the “expansion” notation introduced after Corollary 4.3.) Every color in G_{01} crosses with F . Hence, to avoid a triangle in $G^\#$, there are no cycles in G_{01} , by Lemma 3.2. So G_{01} is a forest.

Suppose that G_{01} consists of more than one component. Let R and S be two

components of G_{01} , and choose u_R in R and u_S in S closest to each other. Since G_1 is an isometric subgraph of G , there is a geodesic P in G_1 between u_R and u_S of length at least two. Note that, by the choice of u_R and u_S , all internal vertices of P are in $G_1 - G_{01}$. Let v_R and v_S be the neighbors in G_{02} of u_R and u_S , respectively, and let Q be a geodesic between v_R and v_S . By Lemma 2.6, P and Q contain the same colors, and each color occurs at most once on P and at most once on Q . So all colors on P and Q cross color F . Let p be the vertex on P adjacent to u_R and q the vertex on Q adjacent to v_R . Since p is in $G_1 - G_{01}$, it follows that p is not adjacent to q . This implies that the edges $u_R \rightarrow p$ and $v_R \rightarrow q$ have different colors. So, by Lemma 3.3, they cross. Moreover, they both cross F , which is impossible. Hence we conclude that G_{01} is connected, so that it is a tree, as is G_{02} .

Assume that there is a color occurring twice in the tree G_{01} . Then choose edges uv and xy of the same color such that on the path $P = u \rightarrow v \rightarrow \dots \rightarrow x \rightarrow y$ in G_{01} all colors on the subpath $u \rightarrow v \rightarrow \dots \rightarrow x$ are distinct. Note that this subpath must be of length at least 3. Then $v \rightarrow \dots \rightarrow P \rightarrow \dots \rightarrow x$ is a geodesic on the one side of the color. Let $u \rightarrow \dots \rightarrow Q \rightarrow \dots \rightarrow y$ be a geodesic on the opposite. By Lemmas 2.6 and 3.3, the color of uv crosses with all other colors on P . Since all colors in G_{01} cross F we would get three mutually crossing colors, which is impossible. So every color in G_{01} occurs exactly once. Hence, by Lemma 2.4, subgraph G_{01} is isometric.

Next we prove that G_{01} is convex. Assume the contrary, and let u, v be vertices in G_{01} , so that there is a u, v -geodesic P , all of whose internal vertices are in $G_1 - G_{01}$. Note that P is of length at least 2. Let Q be the u, v -path in G_{01} , which is, as observed above, also a u, v -geodesic. Then $u \rightarrow \dots \rightarrow P \rightarrow \dots \rightarrow v \rightarrow \dots \rightarrow Q \rightarrow \dots \rightarrow u$ is a cycle. This implies that every color on P is also on Q , and vice versa. Since a cycle contains crossing colors, these must both occur on Q , so that they both cross F as well. Since this is impossible, G_{01} is convex in G .

Similarly, G_{02} is a convex subtree in G .

From the fact that G is bipartite and the sides of all colors are convex, we deduce that G is a median graph (cf. Theorem 1 in [6]). Finally, since three mutually crossing colors in a median graph necessarily force an induced Q_3 , we conclude that G is a cube-free median graph. \square

Theorem 5.1 allows us to characterize several subclasses of partial cubes having nice crossing graphs. The *wheel* W_n consists of the n -cycle C_n together with an extra vertex joined to all the vertices of the cycle; cf. Figure 5.1. The cycle is called the *rim* of the wheel, the extra vertex the *center* of the wheel. The edges incident with the center are the *spokes* of the wheel.

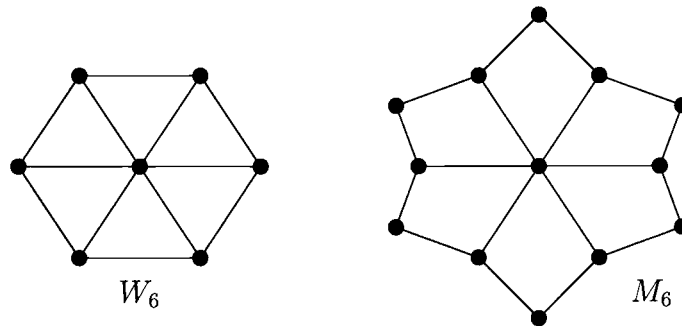
The *cogwheel* M_n is obtained from the wheel W_n by subdividing all the edges on the rim of the wheel; cf. Figure 5.1. Note that the cogwheel M_3 is precisely the cube Q_3 minus a vertex. The *center* and the *spokes* of the cogwheel are inherited from the wheel. The cogwheel M_n is a partial cube with C_n as its crossing graph.

The next proposition is a simple corollary of Theorem 4.4, but it is also straightforward to check it directly.

PROPOSITION 5.2. *Let G be a partial cube. Then $G^\# = K_3$ if and only if G is Q_3 , M_3 , or C_6 .*

THEOREM 5.3. *Let G be a partial cube. Then $G^\#$ is a cycle of length $n \geq 4$ if and only if $G = M_n$.*

Proof. The “if” part of the theorem is obvious. So let $G^\# = C_n$ with $n \geq 4$. By Theorems 3.4 and 5.1, G is a 2-connected cube-free median graph. This implies that the only isometric cycles in G are 4-cycles. Hence any two colors of G cross on some

FIG. 5.1. *The wheel W_6 and the cogwheel M_6 .*

4-cycle. If the side of a color would contain a P_4 or a $K_{1,3}$, then $G^\#$ would contain a vertex of degree at least 3. Hence each side of any color of G must be a P_2 or a P_3 . (Note that P_1 is impossible, since G is 2-connected.) If some side were a P_2 , say of color F , then F would be a pendant vertex in $G^\#$. So we conclude that all sides in G induce a P_3 .

Take a vertex z of maximum degree k in G . Take any edge zu incident with z . If zu is on a 4-cycle with zw , then the colors F_{zu} and F_{zw} cross. On the other hand, color F_{zu} crosses with exactly two other colors. Moreover, F_{zu} crosses with each of these colors on a 4-cycle through z . Hence every edge incident with z is on a 4-cycle with exactly two other edges incident with z . This implies that the colors at z form a 2-regular subgraph of $G^\#$, so that these are all the colors of G . Moreover, the colors cross cycle-wise; that is, we may number the edges incident with z by $0, 1, \dots, n-1$, so that i is exactly on a 4-cycle with edges $i-1$ and $i+1$ modulo k . In the subgraph consisting of all these 4-cycles, all sides already induce a P_3 . So this subgraph comprises all of G , and G is the cogwheel M_n . \square

A C_4 -tree G is recursively defined as follows: G is a 4-cycle, or G is obtained from a C_4 -tree G' by gluing a 4-cycle along an edge to an edge of G' . It is straightforward to prove that G is a C_4 -tree if and only if G can be obtained from two smaller C_4 -trees by gluing them together along an edge (unless $G = C_4$). In [16] it was shown that the central vertices in a C_4 -tree are contained in some 4-cycle or induce a P_4 such that the middle edge of the path is a common edge of two 4-cycles, whereas the first edge is on the one 4-cycle and the last edge is on the other 4-cycle.

Let G be a median graph. Let F be a color of G with split G_1, G_2 . Then we call the color, or the split, *peripheral* if $G_1 = G_{01}$ or $G_2 = G_{02}$. We call the side G_i with $G_i = G_{0i}$ a *peripheral side*. In [26] peripheral colors and sides were called *extremal*. There it was proved that, for any split G_1, G_2 of a median graph, there exists a peripheral split H_1, H_2 such that $H_1 \subseteq G_1$ (and $G_2 \subseteq H_2$).

THEOREM 5.4. *Let G be a graph. Then the following statements are equivalent:*

- (i) G is a partial cube with $G^\#$ a tree.
- (ii) G is obtained from K_2 by successive expansions, where the intersection of the cover is always an edge.
- (iii) G is K_2 or a 2-connected cube-free median graph without induced cogwheels.
- (iv) G is K_2 or a C_4 -tree.

Proof. (i) \Rightarrow (ii) We prove the implication by induction on the number of vertices

in $G^\#$. If $G^\# = K_1$, then $G = K_2$, and we are done. So let $G^\#$ be a nontrivial tree T , and let the color F of G be a vertex of degree 1 in T . By Theorems 3.4 and 5.1, G is a 2-connected cube-free median graph. This implies that the sides of F cannot consist of a single vertex. If the sides contain more than one edge, then F crosses more than one other color of G . So the sides of F consist of a single edge. Let H be the contraction of G with respect to color F . Then $H^\# = T - F$, which is a tree with one vertex less than T . So, by induction, H is obtained by successive expansions, where the intersection of the cover is always a single edge. Hence this also holds for G .

(ii) \Rightarrow (iii) If G is not K_2 , then G is a 2-connected median graph. Cubes can arise only in an expansion when the intersection of the cover contains a 4-cycle. Cogwheels can arise only in an expansion when the intersection of the cover contains a P_3 . So G does not contain Q_3 or any cogwheel.

(iii) \Rightarrow (iv) We use induction on the number of colors in G . If $G = K_2$, then we are done. So let G contain at least two colors. Take a peripheral color F of G , and let $G_1 = G_{01}$ be a peripheral side of F . Since G is a cube-free median graph, we know that G_1 is a tree. Let u be a vertex of degree 1 in G_1 adjacent to w in G_1 . Let v be the neighbor of u in G_{02} and x that of w in G_{02} , so that $C = u \rightarrow v \rightarrow x \rightarrow w \rightarrow u$ is a 4-cycle in G . Note that u is a vertex of degree 2 in G .

If G_1 consists of a single edge, then C is a ‘‘pendant’’ 4-cycle in G (or $G = C$), and we are done by induction. So we may assume that G_1 is a tree with more than two vertices; that is, w has other neighbors in G_1 besides u . Let z be any other neighbor of w in G_1 , and let y be the neighbor of z in G_{02} .

Consider the color F_{uw} . If the sides of F_{uw} consist of a single edge, then the edges uw and vx are on C but not on any other 4-cycle. By deleting the edges uw and vx from G , we obtain two components. Let H_1 be the component containing wx , and let H_2 be the graph obtained from the other component by gluing the 4-cycle C to it along the edge uv . Now G can be obtained from H_1 and H_2 by gluing them together along the edge wx . By induction, H_1 and H_2 are two C_4 -trees. Hence G is one too.

Now consider the case where the sides of F_{uw} consist of more than an edge. Since u is of degree 2, it follows that u is a pendant vertex in the tree S that constitutes the side of F_{uw} in G_{uw} . Let p be any other neighbor of v in the tree S , and let q be its neighbor in G_{vu} . If there is a path between q and y not going through x or u , then let R be a path of minimal length between q and y not going through x or u such that the sum of the distances from x to the vertices on R is as small as possible. Note that R does not use edges of the colors F_{uv} and F_{uw} . We claim that the vertices on R are alternately at distance 2 and 1 from x . If not, then there exists a subpath $r_1 \rightarrow r_2 \rightarrow r_3$ of R with $d(x, r_1) = d(x, r_2) - 1 = d(x, r_3) = k > 1$. The median of x , r_1 , and r_3 is a common neighbor s of r_1 and r_3 at distance $k - 1$ from x . Thus we get another minimal path R' between q and y closer to x , which contradicts the choice of R . Now the vertices x, z, w, v, u, p together with the path R induce a cogwheel in G , which is impossible. Thus we have shown that $\{u, x\}$ is a cutset in G . Let Q_1 and Q_2 be the components of $G - \{u, x\}$, where Q_1 contains p, q, v , and Q_2 contains w, y, z . Let H_1 be the subgraph of G induced by Q_1 and x , and let H_2 be the subgraph of G induced by Q_2 and the vertices x, v , and u . By induction, H_1 and H_2 are C_4 -trees. We can obtain G from H_1 and H_2 by gluing them together along the edge vx ; so G is a C_4 -tree as well.

(iv) \Rightarrow (i) We use induction on the number of 4-cycles that are used to construct G . If G is K_2 or a 4-cycle, then we are done. So assume that more than one 4-cycle

is used to construct G , and let $C = u \rightarrow v \rightarrow x \rightarrow w \rightarrow u$ be the last cycle used in the construction, where the gluing was along the edge uv . Let G' be the graph obtained from G by deleting the vertices x and w together with their incident edges. By induction, G' is a partial cube with a tree T' as its crossing graph. Then G is a partial cube as well, with one extra color. The crossing graph of G is obtained from T' by adding the vertex F_{uw} adjacent to vertex F_{uv} . This completes the proof. \square

Recall that a *block* in a connected graph is a maximal 2-connected subgraph. A C_4 -*cactoid* is a connected graph, each block of which is a K_2 or a C_4 -tree. Loosely speaking, a C_4 -cactoid can be obtained from K_2 's and 4-cycles by gluing them together along vertices or edges.

THEOREM 5.5. *Let G be a graph. Then the following statements are equivalent:*

- (i) G is a partial cube with $G^\#$ a forest.
- (ii) G is obtained from K_2 by successive expansions, where the intersection of the cover is always a vertex or an edge.
- (iii) G is a cube-free median graph without induced cogwheels.
- (iv) G is a C_4 -cactoid.

Proof. For each block of G we may apply Theorem 5.4. Just observe that for statement (ii) we can always add an edge pending at a vertex u to G by an expansion with respect to the cover $G_1 = G$, $G_2 = \langle u \rangle$. Then we can use this edge to construct a new block. \square

6. Crossing graphs and Cartesian products. In this section we consider the relation between crossing graphs and Cartesian products of graphs. As it will turn out, there are several connections involving Cartesian products and joins of graphs. Recall that the *join* $G \oplus H$ of the graphs G and H is the graph obtained from the disjoint union of $G \cup H$ by joining every vertex of G with every vertex of H .

In the previous section we have observed that, if $G^\#$ and $H^\#$ are complete, then so is $(G \square H)^\#$. This fact is a special case of our next result.

PROPOSITION 6.1. *Let G and H be partial cubes. Then $(G \square H)^\# = G^\# \oplus H^\#$.*

Proof. It is easy to see that the Θ -classes of $G \square H$ are in one-to-one correspondence with the union of the Θ -classes of G and the Θ -classes of H . (Instead of proving this fact directly, we refer to Lemma 4.3 of [19].) This means that $V((G \square H)^\#) = V(G^\#) \cup V(H^\#)$. In addition, the Θ -classes of $G \square H$ corresponding to the Θ -classes of G induce $G^\#$, and, analogously, the Θ -classes of $G \square H$ corresponding to the Θ -classes of H induce $H^\#$. Finally, any Θ -class from the induced $G^\#$ crosses with any Θ -class from the induced $H^\#$; in fact, by the definition of the Cartesian product, they cross on a 4-cycle. \square

For instance, Proposition 6.1 implies that the crossing graph of the Cartesian product of n copies of P_3 is the n -octahedron: $(P_3^n)^\# = K_{2,2,\dots,2}$.

Recall from section 3 that $S(G)$ is the simplex graph of a graph G .

PROPOSITION 6.2. *Let G and H be two disjoint graphs. Then $S(G \oplus H) = S(G) \square S(H)$.*

Proof. First note that $S(G)$ and $S(H)$ are subgraphs of $S(G \oplus H)$ having only the vertex \emptyset in common. For any complete subgraph K in $G \oplus H$, let $K_G = K \cap G$ and $K_H = K \cap H$. Note that $V(K)$ is the disjoint union of $V(K_G)$ and $V(K_H)$. Then the mapping ϕ defined by $\phi(K) = (K_G, K_H)$ is a bijection between the vertex set of $S(G \oplus H)$ and the vertex set of $S(G) \square S(H)$. Let K, L be two complete subgraphs in $G \oplus H$. Then K and L are adjacent in $S(G \oplus H)$ if and only if $|K \triangle L| = 1$ if and only if either $|K_H \triangle L_H| = 0$ and $|K_G \triangle L_G| = 1$ or $|K_G \triangle L_G| = 1$ and $|K_H \triangle L_H| = 0$ if and only if (K_G, K_H) and (L_G, L_H) are adjacent in $S(G) \square S(H)$.

So ϕ is an isomorphism. \square

LEMMA 6.3. *Let G be a median graph, and let F_{uv} and F_{uw} be crossing colors. Then $v \rightarrow u \rightarrow w$ is in a 4-cycle.*

Proof. Since F_{uv} and F_{uw} are crossing, there is a vertex y in $G_{vu} \cap G_{wu}$. Then we have

$$d(y, v) = d(y, u) - 1 \quad \text{and} \quad d(y, w) = d(y, u) - 1.$$

Let x be the median of y, v, w ; that is, x is on a geodesic between v and w , on a geodesic between y and v , and on a geodesic between y and w . Then x is a common neighbor of v and w distinct from u , so that $v \rightarrow u \rightarrow w \rightarrow x \rightarrow v$ is a 4-cycle. \square

Let H be a subgraph of a connected graph G , and let z be a vertex of G outside H . A vertex x in H is a *gate* for z in H if, for any vertex w in H , there is a geodesic between z and w passing through x . For the proof of our next theorem we state the following well-known fact.

LEMMA 6.4. *Let H be a subgraph of a connected graph G , and let z be a vertex of $V(G) \setminus V(H)$. Then z has at most one gate in H , which must then be the unique vertex in H closest to z .*

We also recall that it is easy to check that, in a median graph G , every vertex outside a convex subgraph H has a gate in H .

THEOREM 6.5. *Let G be a partial cube. Then $G^\#$ is a complete bipartite graph if and only if G is the Cartesian product of two trees.*

Proof. First let $G = T_1 \square T_2$ be the Cartesian product of two trees T_1 and T_2 . Then the colors of T_i form an independent set in $G^\#$, for $i = 1, 2$, so $G^\#$ is a complete bipartite graph by Proposition 6.1.

Conversely, let $G^\#$ be a complete bipartite graph with bipartition X, Y . Then, by Theorem 5.1, G is a cube-free median graph, so that all sides in G are convex subtrees. In particular, G does not have three mutually crossing colors, and any color occurring in some side occurs only once in that side. Take any color F in G , say between G_1 and G_2 , with sides G_{01} and G_{02} , respectively. Without loss of generality, F is in X . Since each color in Y crosses with F on some 4-cycle, each color in Y occurs in G_{01} as well as G_{02} . Since F does not cross with any other color in X , no color from X occurs in G_{01} or G_{02} .

Similarly, if Φ is any color in Y , then the sides of Φ are convex subtrees of G , in which each color from X occurs exactly once and no color from Y occurs.

Let z be any vertex of G . Note that the existence of three different neighbors of z in $I(u, z)$ would force three mutually crossing colors in G . (Just take the medians of u and any two of these neighbors of z in $I(u, z)$; these produce distinct 4-cycles through z and its three neighbors.) Hence we conclude that there are at most two neighbors of z that are closer to u for any z in G .

Now we are ready to find the appropriate subgraphs in G that will form the factors in the Cartesian product. Let F be a peripheral color with split G_1, G_2 and peripheral side $G_1 = G_{01}$. Without loss of generality, we may assume that F is in X . Let u be a vertex of degree 1 in subtree G_1 with neighbor w in G_1 , and let v be the neighbor of u in G_{02} , so that $F = F_{uv}$ and $G_1 = G_{uv}$. Note that u has degree 2 in G .

First we will show that F_{uw} is a peripheral color with G_{uw} as its peripheral side. Let G'_{uw} be the side of F_{uw} in G_{uw} . Since uw is in G_{uw} and the color F of uw is in X , it follows that G'_{uw} is a tree, in which all colors of X occur exactly once but no color of Y occurs. Assume that some vertex p in G'_{uw} has a neighbor q in $G_{uw} - G'_{uw}$. Then F_{pq} is a color in Y ; so it crosses with all colors in G'_{uw} . By repeatedly applying

Lemma 6.3, we proceed along a path from p to u in G'_{uw} finding an adjacent path in $G_{uw} - G'_{uw}$. Thus we find a neighbor s of u in $G_{uw} - G'_{uw}$, which must be distinct from v and w . This contradicts the fact that u has degree 2. So $G_{uw} = G'_{uw}$, and G_{uw} is a subtree containing all colors of X but no color of Y .

Let $H = G_{uv} \square G_{uw}$. We endow the copies of G_{uv} and G_{uw} in H with the same coloring as G_{uv} and G_{uw} in G , respectively. We will prove that G is isomorphic to H , where the isomorphism preserves the coloring.

Set $|X| = m$ and $|Y| = n$. Then G_{uv} is a tree of size m and order $m + 1$, and G_{uw} is a tree of size n and order $n + 1$. If $G^\# = K_2$, then G is a 4-cycle, and we are done. So we may assume that $G^\# = K_{n,m}$ with $m \geq 2$. First we show that G has the right number of vertices by induction on $n + m$. Then we construct a coordinatization for the product and check adjacencies.

If we delete G_{uv} from G , then we get a partial cube with one less color and with $K_{n,m-1}$ as its crossing graph. So, by induction, we may assume that $G - G_{uv}$ is the Cartesian product of two trees of sizes n and $m - 1$, respectively, so that $G - G_{uv}$ is of order $(n + 1)m$. Since G_{uv} is a tree of size n , it follows that G is of order $(n + 1)m + (n + 1) = (n + 1)(m + 1)$. So G is of the right order.

Let $G_{\leq k}$ be the subgraph of G induced by the vertices of distance at most k to u , and let $H_{\leq k}$ be the subgraph of H induced by the vertices of distance at most k to (u, u) . By induction on k , we will prove the following claim.

Claim. $G_{\leq k} \cong H_{\leq k}$ for $k \geq 0$.

Let z be any vertex of G , let z_w be its gate in G_{uv} , and let z_v be its gate in G_{uw} . Note that $d(z, u) = d(z, z_w) + d(z_w, u) = d(z, z_v) + d(z_v, u)$. We set $z = (z_w, z_v)$. Then $u = (u, u)$; so $G_{\leq 0} \cong H_{\leq 0}$.

Let z be any vertex of G_{uv} . Then we have $z = (z, u)$. Since G_{uv} is a convex subtree of G , there is a unique neighbor y of z closer to u , and y is the neighbor of z on the path from z to u in the tree G_{uv} . Then we have $y = (y, u)$. This implies that the subgraph G_{uv} of G is isomorphic to the subgraph $G_{uv} \square \{u\}$ of H . Similarly, the subgraph G_{uw} of G is isomorphic to the subgraph $\{u\} \square G_{uw}$ of H . In particular, we have shown that the claim is true for $k \leq 1$.

Now let z be a vertex of G outside $G_{uv} \cup G_{uw}$ with $d(u, z) = k$. Then we have $d(z, z_w), d(z_w, u), d(z, z_v), d(z_v, u) \geq 1$, so that $k \geq 2$. Let p be a neighbor of z on a geodesic from z to z_w . Then we have $p_w = z_w$, so that $p = (z_w, p_v)$. Moreover, we have $d(p, u) = d(p, z_w) + d(z_w, u) = d(z, u) - 1 = k - 1$. By induction, we know the following facts. There is a unique geodesic P between p and z_w of length $d(p, z_w) = d(p_v, u)$, of which all the colors are in X . There is a unique geodesic Q between p and p_v of length $d(p, p_v) = d(z_w, u)$, of which all the colors are in Y .

Since $z \rightarrow P$ is a geodesic between z and its gate z_w in G_{uv} and all colors of Y occur in G_{uv} , color F_{z_p} must be in X . Hence F_{z_p} crosses with every color on Q . So, by repeatedly applying Lemma 6.3, we can construct a path Q' along Q from z to a neighbor r of p_v of the same length and coloring as Q . Since the last color on Q is F_{uv} , the last color on Q' is also F_{uv} , so that r is in G_{uv} . By the unicity of gates, we have $z_v = r$. Hence p_v is the unique neighbor of z_v in subtree G_{uw} closer to u . Let q be the neighbor of z on Q' . By a similar argument, we deduce that q_w is the unique neighbor of z_w in subtree G_{uv} closer to u .

Now $p = (z_w, p_v)$ and $q = (q_w, z_v)$ are two distinct neighbors of z closer to u . Hence these are all neighbors of z closer to u in G . This settles the induction step in the proof of the claim.

TABLE 7.1
Summary of the results of the paper.

$G^\#$	G
connected	2-connected
edgeless	tree
complete	obtained by all-color expansion
triangle-free	cube-free median
K_3	$Q_3, M_3, \text{ or } C_6$
$C_n, n \geq 4$	M_n
tree	K_2 or C_4 -tree
forest	C_4 -cactoid
complete bipartite	Cartesian product of two trees

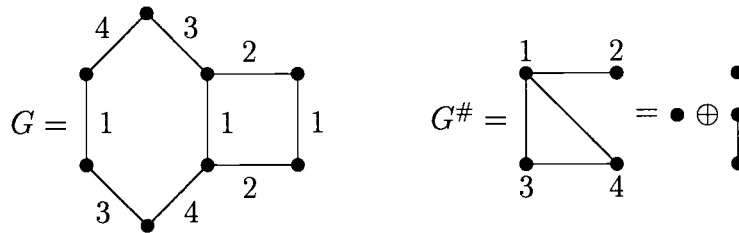


FIG. 7.1. A partial cube and its crossing graph.

Since G and H have the same number of vertices, we infer that

$$G = G_k \cong H_k = H,$$

for $k = \text{diameter}(G)$, by which the proof is complete. \square

7. Concluding remarks. Most of the results of this paper can be summarized in Table 7.1.

The last entry in the table, that is, Theorem 6.5, raises the following question.

PROBLEM 7.1. *What can be said about the partial cube G if its crossing graph $G^\#$ is the join of two other graphs that are not edgeless?*

This seems to be a tough problem as the examples in Figures 7.1 and 7.2 show. The graph in Figure 7.1 is a partial cube but not a median graph, whereas its crossing graph is still the join of two smaller graphs. The graph in Figure 7.2 is a median graph but not the Cartesian product of two smaller graphs, whereas its crossing graph is still the join of two smaller graphs.

One may define an equivalence relation $\kappa_\#$ on the family of all partial cubes as follows: two partial cubes are in relation $\kappa_\#$ to each other if they have isomorphic crossing graphs. Theorem 6.5 and Proposition 4.4 may be considered as instances of the characterization of two of the equivalence classes of this relation. A related problem is the following.

PROBLEM 7.2. *Determine all C_4 -trees having the same tree as crossing graph.*

Finally, Theorem 5.3 suggests the following question for a median graph G .

PROBLEM 7.3. *Does an induced cycle C_n in $G^\#$ necessarily force an induced cogwheel M_n in G ?*

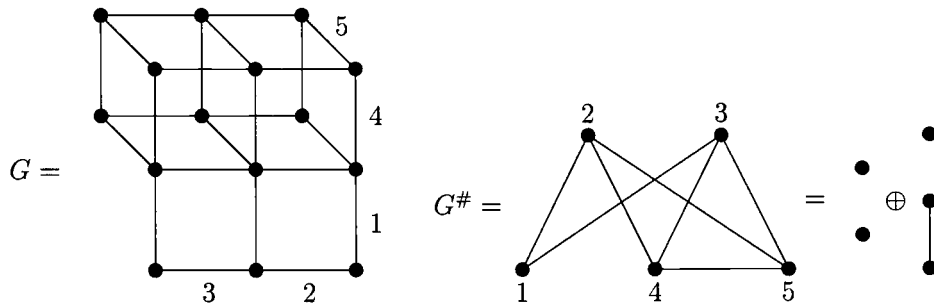


FIG. 7.2. A median graph and its crossing graph.

REFERENCES

- [1] F. AURENHAMMER, M. FORMANN, R. IDURY, A. SCHÄFFER AND F. WAGNER, *Faster isometric embeddings in products of complete graphs*, Discrete Appl. Math., 52 (1994), pp. 17–28.
- [2] F. AURENHAMMER AND J. HAGAUER, *Recognizing binary Hamming graphs in $O(n^2 \log n)$ time*, in Proceedings of the 16th International Workshop on Graph Theoretical Concepts in Computer Science, Lecture Notes in Comput. Sci. 484, Springer-Verlag, New York, 1991, pp. 90–98.
- [3] F. AURENHAMMER AND J. HAGAUER, *Recognizing binary Hamming graphs in $O(n^2 \log n)$ time*, Math. Systems Theory, 28 (1995), pp. 387–395.
- [4] D. AVIS, *Hypermetric spaces and the Hamming cone*, Canad. J. Math., 33 (1981), pp. 795–802.
- [5] H.-J. BANDELT AND J.-P. BARTÉLEMY, *Medians in median graphs*, Discrete Appl. Math., 8 (1984), pp. 131–142.
- [6] H.-J. BANDELT, H. M. MULDER, AND E. WILKEIT, *Quasi-median graphs and algebras*, J. Graph Theory, 18 (1994), pp. 681–703.
- [7] H.-J. BANDELT AND M. VAN DE VEL, *Superextensions and the depth of median graphs*, J. Combin. Theory Ser. A, 57 (1991), pp. 187–202.
- [8] V. D. CHEPOI, *d -Convexity and isometric subgraphs of Hamming graphs*, Cybernetics, 1 (1988), pp. 6–9.
- [9] V. D. CHEPOI AND S. KLAVŽAR, *The Wiener index and the Szeged index of benzenoid systems in linear time*, J. Chem. Inform. Comput. Sci., 37 (1997), pp. 752–755.
- [10] D. DJOKOVIĆ, *Distance preserving subgraphs of hypercubes*, J. Combinatorial Theory Ser. B, 14 (1973), pp. 263–267.
- [11] K. FUKUDA AND K. HANDA, *Antipodal graphs and oriented matroids*, Discrete Math., 111 (1993), pp. 245–256.
- [12] M. R. GAREY AND R. L. GRAHAM, *On cubical graphs*, J. Combinatorial Theory Ser. B, 18 (1975), pp. 84–95.
- [13] R. L. GRAHAM AND H. POLLAK, *On the addressing problem for loop switching*, Bell System Tech. J., 50 (1971), pp. 2495–2519.
- [14] R. L. GRAHAM AND P.M. WINKLER, *On isometric embeddings of graphs*, Trans. Amer. Math. Soc., 288 (1985), pp. 527–536.
- [15] J. HAGAUER, W. IMRICH, AND S. KLAVŽAR, *Recognizing median graphs in subquadratic time*, Theoret. Comput. Sci., 215 (1999), pp. 123–136.
- [16] S. M. HEDETNIEMI, S. T. HEDETNIEMI, AND P. J. SLATER, *Centers and medians of $C_{(n)}$ -trees*, Utilitas Math., 21 (1982), pp. 225–234.
- [17] W. IMRICH AND S. KLAVŽAR, *A simple $O(mn)$ algorithm for recognizing Hamming graphs*, Bull. Inst. Combin. Appl., 9 (1993), pp. 45–56.
- [18] W. IMRICH AND S. KLAVŽAR, *A convexity lemma and expansion procedures for bipartite graphs*, European J. Combin., 19 (1998), pp. 677–685.
- [19] W. IMRICH AND S. KLAVŽAR, *Product Graphs: Structure and Recognition*, John Wiley & Sons, New York, 2000.
- [20] W. IMRICH, S. KLAVŽAR, AND H. M. MULDER, *Median graphs and triangle-free graphs*, SIAM J. Discrete Math., 12 (1999), pp. 111–118.
- [21] S. KLAVŽAR AND I. GUTMAN, *Wiener number of vertex-weighted graphs and a chemical appli-*

- cation*, Discrete Appl. Math., 80 (1997), pp. 73–81.
- [22] S. KLAVŽAR AND H. M. MULDER, *Median graphs: Characterizations, location theory and related structures*, J. Combin. Math. Combin. Comput., 30 (1999), pp. 103–127.
- [23] F. R. MCMORRIS, H. M. MULDER, AND F. R. ROBERTS, *The median procedure on median graphs*, Discrete Math., 84 (1998), pp. 165–181.
- [24] H. M. MULDER, *The structure of median graphs*, Discrete Math., 24 (1978), pp. 197–204.
- [25] H. M. MULDER, *The Interval Function of a Graph*, Math. Centre Tracts 132, Mathematisch Centrum, Amsterdam, 1980.
- [26] H. M. MULDER, *The expansion procedure for graphs*, in Contemporary Methods in Graph Theory, R. Bodendiek, ed., B.I.-Wissenschaftsverlag, Mannheim, Wien, Zürich, 1990, pp. 459–477.
- [27] E. WILKEIT, *Isometric embeddings in Hamming graphs*, J. Combin. Theory Ser. B, 50 (1990), pp. 179–197.
- [28] P. WINKLER, *Isometric embeddings in products of complete graphs*, Discrete Appl. Math., 7 (1984), pp. 221–225.

EXACT SIZE OF BINARY SPACE PARTITIONINGS AND IMPROVED RECTANGLE TILING ALGORITHMS*

PIOTR BERMAN[†], BHASKAR DASGUPTA[‡], AND S. MUTHUKRISHNAN[§]

Abstract. We prove the following upper and lower bounds on the exact size of binary space partition (BSP) trees for a set of n isothetic rectangles in the plane:

- An upper bound of $3n - 1$ in general, and an upper bound of $2n - 1$ if the rectangles tile the underlying space. This improves the upper bounds of $4n$ in [V. Hai Nguyen and P. Widmayer, *Binary Space Partitions for Sets of Hyperrectangles*, Lecture Notes in Comput. Sci. 1023, Springer-Verlag, Berlin, 1995; F. d’Amore and P. G. Franciosa, *Inform. Process. Lett.*, 44 (1992), pp. 255–259]. A BSP satisfying the upper bounds can be constructed in $O(n \log n)$ time.
- A worst-case lower bound of $2n - o(n)$ in general, and $\frac{3n}{2} - o(n)$ if the rectangles form a tiling.

The BSP tree is one of the most popular data structures in computational geometry, and hence even “small” factor improvements of $\frac{4}{3}$ or 2 on the previously known upper bounds that we show improve the performances of applications relying on the BSP tree. As an illustration, we present improved approximation algorithms for certain dual rectangle tiling problems using our upper bounds on the size of the BSP trees.

Key words. binary space partitions, exact bounds, tiling, approximation algorithms

AMS subject classifications. 68Q01, 68W25, 68W40

PII. S0895480101384347

1. Introduction. Binary space partitioning (BSP) for a collection of geometric objects in the two-dimensional plane¹ is defined as follows. The plane is divided into two parts by cutting objects with a line if necessary. Each fragment of the object belongs solely to one of the parts it falls in. The two resulting parts of the plane are divided recursively in a similar manner; the process continues until at most one two-dimensional fragment of the original objects remains in any part of the plane.² This division process can be naturally represented as a binary tree (BSP tree) where a node represents a part of the plane and stores the cut that splits the plane into two parts that its two children represent; each leaf of the BSP tree represents the final partitioning of the plane and stores at most one fragment of an input object. Since a cut at some node may split an object into two, the number of regions in the final configuration, equivalently the number of leaves in the BSP tree, may exceed the number of input objects. Note that quadtrees, octtrees, and grid files are all related to BSPs. Figure 1.1 shows a binary space partition for a set of four rectangles and the corresponding tree.

*Received by the editors January 30, 2001; accepted for publication (in revised form) January 15, 2002; published electronically April 8, 2002.

<http://www.siam.org/journals/sidma/15-2/38434.html>

[†]Department of Computer Science, Pennsylvania State University, University Park, PA 16802 (berman@cse.psu.edu). This author’s research was supported in part by NSF grant CCR-9700053 and by NLM grant LM05110.

[‡]Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053 (dasgupta@cs.uic.edu). This author’s research was supported in part by NSF grants CCR-9800086 and CCR-0296041.

[§]AT& T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932 (muthu@research.att.com).

¹We restrict ourselves to the two-dimensional plane throughout this paper.

²The objects have to be *disjoint*, since otherwise no BSP exists in which each final part of the plane contains at most one fragment of an object.

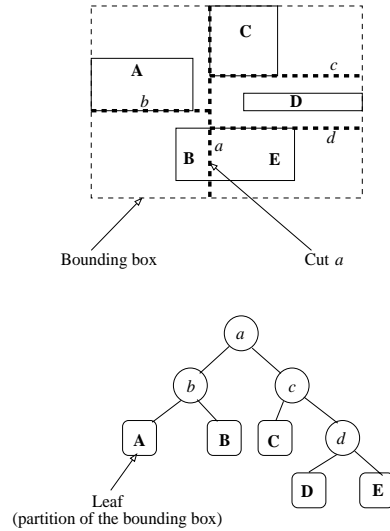


FIG. 1.1. A binary space partition for four isothetic rectangles and the corresponding BSP tree. a , b , c , and d denote the four cuts in the partition, whereas A , B , C , D , and E denote the partitions of the bounding box of these rectangles generated by these cuts such that at most one fragment of each given rectangle is in each such partition.

Since the introduction of BSP trees in [FKN80], they have become one of the most popular data structures. They present a way to implement a geometric divide-and-conquer strategy. They have found numerous applications in graphics (hidden surface removal [D94], shadow generation [CF89]), computational geometry (ray-tracing [NT86], visibility problems [T92], solid geometry [TN87]), robotics (motion planning [B93]), spatial databases [S90, van90] and approximation algorithms [KMP98, M90]. They are used in computer games such as DOOM and Quake, and there is a lot of publicly available code for different kinds of BSPs.

The fundamental parameter of interest about BSPs in all their applications is the *size*, that is, the number of leaves of a BSP tree. In two seminal papers, Paterson and Yao [PY90, PY92] established the first and essentially optimal bounds on the size of some BSPs. In two dimensions, which is of interest to us, they proved that any set of n line segments has a BSP of size $O(n \log n)$; for axis-parallel line segments, they proved that BSPs of $O(n)$ size exist. Very recently, Tóth [T01] showed that there exists n disjoint line segments in the plane such that any BSP of these segments must have a size of at least $\Omega(\frac{n \log n}{\log \log n})$. Some results are in [deGO] for some more general class of objects on the plane. The problem of bounding the size of BSPs remains an active research area, e.g., in higher dimensions, for objects with bounded aspect ratio, etc. [NW95, AG+96].

In this paper, we focus on axis-parallel (isothetic) rectangles as do Paterson and Yao in [PY92]. Such rectangles form a very important class of objects in application domains because complex objects are often replaced by their bounding rect-

angles. They also arise naturally in planar tiling problems such as constructing two-dimensional histograms and as subproblems when higher-dimensional hyperrectangles are projected on two dimensions. Our focus is on proving bounds on the *exact* size of BSP trees in the worst case, that is, not just asymptotic size but also the exact constants involved. In this paper, we prove the following results:

1. There exists a BSP tree of size at most $3n - 1$ for collection of n isothetic rectangles. If the rectangles form a tiling (that is, the given set of rectangles partition a rectangular region), then we prove an improved upper bound of $2n - 1$. A BSP satisfying the upper bounds can be constructed in $O(n \log n)$ time in either case.

Paterson and Yao proved an upper bound of $12n$ in [PY92]; subsequent improvements have led to the current best upper bound of $4n$ [NW95, dAF92].

2. We also present lower bounds on the size of a BSP tree for n isothetic rectangles in the worst case: we prove a lower bound of $2n - o(n)$. If the rectangles must form a tiling of the space, we show a lower bound of $\frac{3n}{2} - o(n)$.

Size of the BSP trees determines the time and space of all applications reliant on them; improvements even by “small” factors ($\frac{4}{3}$ or 2) are highly desirable. Even construction time in securing these improvements is not a bottleneck because our upper bounds can be achieved by efficient algorithms taking $O(n \log n)$ time. Our constructions are modifications of the original algorithm in [PY92], matching it in their running times. The bulk of our technical achievement is our detailed amortized analyses of the algorithm.

Concurrent as well as subsequent to our work, there has been considerable interest in proving tight lower and upper bounds on the exact size of the BSP tree for various objects. In particular, subsequent to our initial submission of this manuscript, Dumitrescu, Mitchell, and Sharir [DMM01] have proved an asymptotic lower bound of $2n - o(n)$ for BSPs for n orthogonal line segments and an improved lower bound of $\frac{9n}{4} - o(n)$ for sizes of BSPs for n isothetic rectangles.³ However, this last lower bound of [DMM01] does not seem to apply to the case when the rectangles must form a tiling of the underlying space.

We show an application of our results for the BSP tree. The dual tiling problem is to cover a two-dimensional array with nonoverlapping rectangles so that the total (or maximum) “weight” of the tiling is bounded by a specified threshold. (The weight of a tiling is determined by different applications in spatial data structures, databases, parallel load balancing, video compression, etc.) The goal is to minimize the number of rectangles used in the tiling. Using our upper bounds on the BSP trees for isothetic rectangles, we present approximations for the dual tiling problem which significantly improves known results in approximation factor and/or running time. This technique is general and applies to different “weight” functions and optimization criteria.

Map. In the rest of the paper, we first present the upper bounds (section 2) followed by the lower bounds (section 3). We then present applications of our results to the dual rectangle tiling problems in section 4.

2. Upper bounds. Given a rectangular region \mathbf{R} containing a set of n disjoint isothetic rectangles, a BSP of \mathbf{R} consists of recursively partitioning \mathbf{R} by a horizontal or vertical line into two subregions and continuing in this manner for each of the two subregions until each obtained region intersects at most one rectangle. If a rectangle

³They also announced in the same conference that the lower bound of $\frac{9n}{4} - o(n)$ can be further improved to $\frac{7n}{3} - o(n)$.

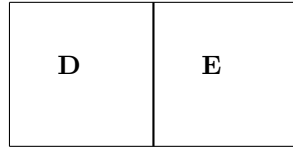


FIG. 2.1. A cut of \mathbf{C} .

is intersected by a cutting line, it is then split into disjoint rectangles whose union is the intersected rectangle. The *size* of a BSP is the number of regions produced. Equivalently, a BSP of a set of rectangles contained in a rectangular region \mathbf{R} is a binary tree, where each node is a rectangular subregion of \mathbf{R} , each internal node is the union of its two children, the intersection of each leaf with the rectangles in our collection has at most one rectangle, and the root is the rectangular region \mathbf{R} . The partition of \mathbf{R} that corresponds to a BSP is the collection of the leaves of this tree, and the size of the BSP is the number of leaves in the tree. A set of rectangles form a tiling if they partition some rectangular region \mathbf{R} . It is shown in [KMP98, MPS99] that a BSP of \mathbf{R} with the minimum number of leaves can be computed using dynamic programming technique in $O(n^5)$ time.

THEOREM 2.1. *We can compute a BSP of \mathbf{R} of size at most $3n - 1$ in $O(n \log n)$ time.*

Restricting the set of rectangles in our collection to be a tiling of \mathbf{R} yields even better bounds on the size of the BSP.

THEOREM 2.2. *Assume that the rectangles in our collection form a tiling of \mathbf{R} . Then, we can compute a BSP of \mathbf{R} of size at most $2n - 1$ in $O(n \log n)$ time.*

In the rest of this section, we will prove both theorems.

2.1. General schema for the proofs of Theorems 2.1 and 2.2. Both Theorems 2.1 and 2.2 use the accounting method of amortized analysis for tighter bounds on the size of their respective BSPs. First, we describe the general schema and then later provide details of how this schema can be applied to each case.

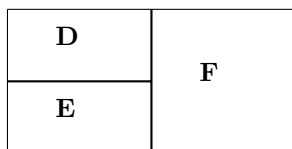
Suppose that a node (region) \mathbf{C} in a BSP has two children, say \mathbf{D} and \mathbf{E} . Then the boundary between \mathbf{D} and \mathbf{E} is a line segment, horizontal or vertical, that extends between two points on the boundary of \mathbf{C} . This segment is called a *cut* of \mathbf{C} (see Figure 2.1).

Our proof is by induction. We use \mathcal{N} to denote the given collection of the disjoint rectangles. For a subrectangle of \mathbf{R} , say \mathbf{C} , we define induced collection $\mathcal{N}_{\mathbf{C}}$, which consists of nonempty intersections of rectangles from \mathcal{N} with \mathbf{C} .

We start by declaring one of the sides of \mathbf{R} to be *unbroken* and the others to be *broken*. Then we initially give money to the elements of $\mathcal{N} = \mathcal{N}_{\mathbf{R}}$ according to the following formula (for some x and y , with $x \geq y \geq 1$): a rectangle not adjacent to a broken side gets x dollars, a rectangle adjacent to only one broken side gets y dollars, and any other rectangle (adjacent to two or more broken sides) gets one dollar. We will say that $\mathcal{N}_{\mathbf{R}}$ is *properly endowed* if the associated distribution of dollars satisfies this formula. The choice of x and y will vary for the two problems.

Our inductive step is the following. If a rectangle \mathbf{C} has three broken sides and $\mathcal{N}_{\mathbf{C}}$ is properly endowed, then

- either $|\mathcal{N}_{\mathbf{C}}| = 1$;
- or we can cut \mathbf{C} into two rectangles \mathbf{D} and \mathbf{E} , declare three broken sides for these rectangles, and provide proper endowment for $\mathcal{N}_{\mathbf{D}}$ and $\mathcal{N}_{\mathbf{E}}$ by re-

FIG. 2.2. Cutting \mathbf{C} twice into \mathbf{D} , \mathbf{E} , and \mathbf{F} .

distributing enough of the money from the proper endowment of $\mathcal{N}_{\mathbf{C}}$ (see Figure 2.1);

- or we can cut \mathbf{C} twice, into three rectangles \mathbf{D} , \mathbf{E} , and \mathbf{F} (see Figure 2.2), and make the declarations of broken sides and redistribution of dollars for all three new rectangles (one child and two grandchildren of \mathbf{C}).

This inductive argument is sufficient (with $x = 3$ and $y = \frac{3}{2}$ for Theorem 2.1 and with $x = 2$ and $y = 1$ for Theorem 2.2), because every leaf in the resulting BSP gets at least one dollar, and initially there are at most $xn - 1$ dollars; thus we have less than $xn - 1$ leaves in our BSP since we can obviously assume that the root \mathbf{R} is the smallest bounding rectangle of the given set of rectangles.⁴ Finally, it is easy to eliminate empty regions in the resulting BSP, if so desired, by ensuring that each region intersects at least one rectangle of our collection: if the region for a nonroot node does not intersect any rectangle, then simply cover it by extending its sibling in the BSP and, if necessary, the descendants of this sibling.

The inductive step is easy if we can cut \mathbf{C} without splitting any of the rectangles in $\mathcal{N}_{\mathbf{C}}$, i.e., we have $\mathcal{N}_{\mathbf{C}} = \mathcal{N}_{\mathbf{D}} \cup \mathcal{N}_{\mathbf{E}}$, $\mathcal{N}_{\mathbf{C}} \neq \mathcal{N}_{\mathbf{D}}$, and $\mathcal{N}_{\mathbf{C}} \neq \mathcal{N}_{\mathbf{E}}$. A side of the child rectangle that is a (part of a) broken side of \mathbf{C} is declared broken. If a child has two undeclared sides, we arbitrarily declare one of them to be broken. Since $x \geq y \geq 1$, it is easy to see that no rectangle from $\mathcal{N}_{\mathbf{C}}$ needs more money after this partition. We refer to this case as the *easy case* of the inductive step.

For ease of discussion, we assume that the unbroken side of \mathbf{C} is the right side of \mathbf{C} (by rotating the coordinate axes, if necessary).

2.2. Application of the general schema. Now, we give details of the application of the general schema for each of the theorems.

2.2.1. Proof of Theorem 2.1. Set $x = 3$ and $y = 3/2$. Our algorithm is a modification of the algorithm discussed in [PY92]. Refer to Figure 2.3. Let X be the longest rectangle which is adjacent to the broken left side of \mathbf{C} , or, in the absence of any such rectangle, the rectangle in \mathbf{C} whose left side is closest to the left side of \mathbf{C} (break ties arbitrarily). Slide the right vertical side (segment) $\overline{a, b}$ of X until it either hits a rectangle in \mathbf{C} or hits the unbroken side of \mathbf{C} .⁵ If $\overline{a, b}$ hits the unbroken side of \mathbf{C} , then this is the easy case of the inductive step. Otherwise, let $\overline{c, d}$ be the position of $\overline{a, b}$ when it hits a rectangle Y of \mathbf{C} . Notice that if Y is adjacent to two broken (horizontal) sides of \mathbf{C} , then we have the easy case of the inductive step; hence we assume that this is not the case.

Our first cut is vertical through point d . If this cut does not split any rectangles of $\mathcal{N}_{\mathbf{C}}$, we have the easy case. Otherwise, we will assume that this cut splits a rectangle above line $\overline{b, d}$. The sides of \mathbf{D} , \mathbf{E} , and \mathbf{F} contributed by the first cut are

⁴It is easy to see that in the beginning at least one rectangle will be adjacent to at least one broken side.

⁵The notation $\overline{a, b}$ denote the segment joining points a and b .

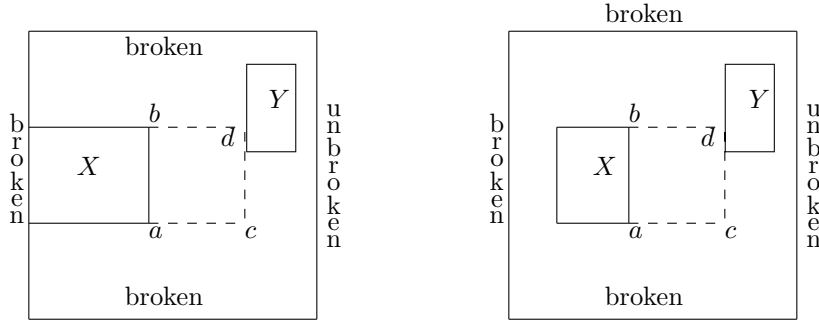


FIG. 2.3. The placement $\overline{c, d}$ of slided $\overline{a, b}$.

declared broken. (Rectangles **D**, **E**, and **F** were defined in the discussion of the general schema.) Our second cut will be the horizontal line $\overline{b, d}$ extended to the left side of **C**. The sides of children rectangles that are adjacent to that line are declared unbroken. The second cut is the easy case of the inductive step in the sense that it does not need to be analyzed in terms of its effects on the endowment but was essential because it provided the left products of the cuts with unbroken sides. While checking if we have enough money for the endowments, we analyze the first cut only. It splits a number of rectangles in **C**, among which at most two are adjacent to one broken side of **C** (one adjacent to the top side and the other to the bottom side of **C**). Every other rectangle that is cut had three dollars before the cut, and after the cut each of the two pieces needs $\frac{3}{2}$ dollars, which can be obtained by distributing the three dollars evenly between them. Hence, there are two cases to consider:

- Only one rectangle adjacent to a broken side of **C** is cut. Before the cut, the rectangle had $\frac{3}{2}$ dollars. After the cut, each of the two pieces need one dollar; hence it needs an extra $\frac{1}{2}$ dollar for proper endowment. However, because the status of rectangle **Y** is changed (that is, because **Y** becomes adjacent to at least one more broken side), it has a surplus of at least $\min\{x - y, y - 1\} \geq \frac{1}{2}$ dollars, which can be used for this purpose.
- Two rectangles adjacent to a broken side of **C** are cut. Then, each of these rectangles needs an additional $\frac{1}{2}$ dollar. However, in this case, **Y** could not have been adjacent to any broken side of **C** before the cut but becomes adjacent to one broken side after the cut; hence it has a surplus of at least $x - y = \frac{3}{2}$ dollars, which can be used to provide the additional one dollar.

We now turn to the implementation of our algorithm. All the steps of our algorithm can be implemented in a manner similar to those in [dAF92], except the step that requires finding the placement $\overline{c, d}$ of the slided vertical side $\overline{a, b}$ of **X** if it was stopped by the left vertical side of some rectangle **Y**. We describe the implementation, assuming that the side $\overline{a, b}$ is vertical and the unbroken side of **C** is the right side of **C**; the other cases are similar. The x and y coordinates of the corners of the given rectangles can be preprocessed in $O(n \log n)$ time by sorting them and then replacing them by their rank in the respective sorted lists to ensure that each coordinate of any corner point of any rectangle is an integer from the set $\{1, 2, \dots, 2n\}$. Whenever we generate a new rectangular partition, we also maintain the coordinates of its top left and bottom right corners. We now need to maintain a data structure on the left vertical sides of all rectangles in our collection **R** such that, given a query vertical segment (which is a part of the right vertical segment of some rectangle in our collection **R**) connecting

two points with coordinates (x, y_0) and (x, y_1) , the query returns the left vertical side of a rectangle connecting points with coordinates (x', y'_0) and (x', y'_1) such that the interval $[y_0, y_1]$ overlaps the interval $[y'_0, y'_1]$, $x' \geq x$ and x' is the minimum possible; then we need to check if the segment connecting the points (x', y_0) and (x', y_1) is inside our current bounding rectangular region \mathbf{C} using the coordinates of its top left and bottom right corners of \mathbf{C} . (Otherwise, the segment $\overline{a, b}$, when slid to the right, would hit the right unbroken side of \mathbf{C} .)

Therefore, it suffices for our purpose to build a data structure \mathcal{D} on a set of n two-dimensional points $\{(x, y) \mid x, y \in \{1, 2, \dots, 2n\}\}$ in $O(n \log n)$ time and using $O(n \log n)$ space such that a query range (a, b, c) must return a point in \mathcal{D} (if any) in $O(\log n)$ time with $y \leq b \leq c$, $a \geq x$ and a is the minimum possible.⁶ We modify the priority search tree (PST) data structure (e.g., see [M85]) for this purpose. A PST is a data structure on n two-dimensional points such that given a query range (a, b, c) it is possible to find in $O(\log n)$ time a point (x, y) in the PST with $x \geq a$ (or, $x \leq a$) and $b \leq y \leq c$; the first dimension is called the priority dimension and the remaining two dimensions are called the search dimensions. Moreover, it is possible to build a PST in $O(n \log n)$ time using $O(n)$ space. To design \mathcal{D} , we first build a binary search tree T on the y -coordinates of the given points. At each node v we store a horizontal line H_v which represents a value in between the maximum y -coordinate value in the left subtree and the minimum y -coordinate value in the right subtree. At the left (respectively, right) child of v we store an appropriate PST whose search dimension is based on the x -coordinate values and whose priority dimension is based on the y -coordinate values of the given points. Given a query range (x, y_0, y_1) with the first dimension being the priority dimension, we search in T and find the highest node v such that H_v is contained in $[y_0, y_1]$. H_v splits the range into two subranges that are both unbounded with respect to the priority dimension. (For example, the part of the range below H_v is unbounded in the negative y -dimension and similarly for the other range.) So, we simply use the PSTs to find the point in each subrange which has the minimum value in the search dimension (i.e., the x -dimension). From these two queries we can infer our desired point. We do this query twice at a single node v , after searching down T . So the query time is $O(\log n)$. The storage per level of T is $O(n)$, so it is $O(n \log n)$ overall. If we are given the points in sorted order along the search dimension, then a PST on those points can be built in linear time. Thus, we can presort all the points once and then build the entire structure bottom-up (merging the sorted lists at the left and right child of v to get the sorted list at v). So the total preprocessing time is also $O(n \log n)$.

2.2.2. Proof of Theorem 2.2. Set $x = 2$ and $y = 1$. For a subrectangle \mathbf{U} of \mathbf{C} let $\mathcal{B}_{\mathbf{U}}$ to be the union of the boundary segments (sides) of the rectangles of $\mathcal{N}_{\mathbf{U}}$ that are not contained in the boundary of \mathbf{U} . We pick a horizontal line segment $\overline{a, b}$ with the following properties (see Figure 2.4):

- a belongs to the left side of \mathbf{C} (remember that the left vertical side of \mathbf{C} is broken);
- $\overline{a, b}$ is a horizontal line segment;
- $\overline{a, b}$ is a subset of $\mathcal{B}_{\mathbf{C}}$ (that is, $\overline{a, b}$ consists of line segments each of which is an element of $\mathcal{B}_{\mathbf{C}}$);
- no point on $\overline{a, b}$, except possibly the points a and b , lie on a side of \mathbf{C} ;
- $\overline{a, b}$ is a *longest* segment with the above properties.

⁶We would like to thank Prof. Ravi Janardan for helping us with this part of the implementation.

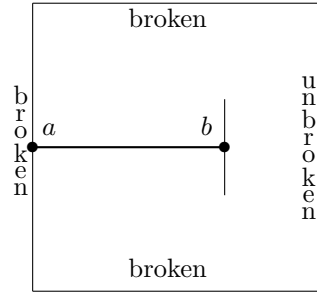


FIG. 2.4. Picking the horizontal line segment $\overline{a, b}$.

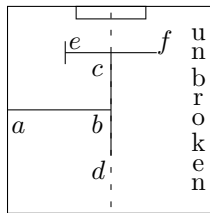


FIG. 2.5. Extending $\overline{b, c}$ upwards towards the boundary of \mathbf{C} .

Assume that b is not on a side of \mathbf{C} ; otherwise we would have the easy case as described before. Also, if there is no such segment $\overline{a, b}$ of length greater than zero, then again we would have the easy case, so we may assume that such a segment $\overline{a, b}$ of length greater than zero exists. Our first cut is vertical through point b . The sides of \mathbf{D} , \mathbf{E} , and \mathbf{F} contributed by this cut are declared broken. The second cut is $\overline{a, b}$ itself, and the sides of \mathbf{D} and \mathbf{E} contributed by this cut are declared unbroken. As before, any side of \mathbf{D} , \mathbf{E} , and \mathbf{F} that is part of a broken side of \mathbf{C} is also declared broken. It remains to show that the endowment of $\mathcal{N}_{\mathbf{C}}$ is sufficient to provide the endowments of $\mathcal{N}_{\mathbf{D}}$, $\mathcal{N}_{\mathbf{E}}$, and $\mathcal{N}_{\mathbf{F}}$.

Since the given collection of rectangles form a tiling, point b must lie in the interior of a vertical segment from $\mathcal{B}_{\mathbf{C}}$ that is perpendicular to $\overline{a, b}$. We extend this segment maximally vertically in both directions, so it is a union of segments $\overline{c, b}$ and $\overline{b, d}$ (see Figure 2.5). If both c and d are on the boundary of \mathbf{C} , then the cut $\overline{c, d}$ makes it an easy case. Thus we can assume that at least one of them, say c , is located in the interior of \mathbf{C} and thus in the interior of some maximal horizontal segment (i.e., a horizontal segment that cannot be extended in either direction) from $\mathcal{B}_{\mathbf{C}}$, say e, f . Assume that e is to the left of f .

Observe that e must be located in the interior of \mathbf{C} , since otherwise the segment e, f would be chosen, rather than $\overline{a, b}$. Thus, since the given collection of rectangles form a tiling, the rectangle of $\mathcal{N}_{\mathbf{C}}$ that is adjacent to segments $\overline{e, c}$ and $\overline{b, c}$ is not adjacent to the boundary of \mathbf{C} , and therefore it was endowed with two dollars. During our cuts this rectangle is not subdivided, and it is adjacent to our new broken side; thus we can take one dollar from its endowment to be used elsewhere. To perform the first cut, we extend $\overline{b, c}$ upwards toward the boundary of \mathbf{C} . We may cut a number of rectangles from $\mathcal{N}_{\mathbf{C}}$. The last one is adjacent to a broken boundary segment of \mathbf{C} , so it has only one dollar, and after the cut we need one dollar for each of its two

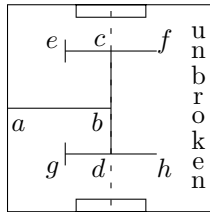


FIG. 2.6. Extending $\overline{b, d}$ downwards towards the boundary of \mathbf{C} .

parts. This is where we used a dollar that we have extracted a moment ago. Other rectangles which are cut cannot be adjacent to a broken boundary, since the only possibility is if the rectangle is adjacent to the boundary segment where a is located, but then such a rectangle would provide a better (longer) choice for $\overline{a, b}$. Thus each of these rectangles has two dollars, but after the cut their parts are adjacent to the new broken boundary segment, so they need one dollar each.

Now we extend $\overline{b, d}$ downwards toward the boundary of \mathbf{C} (see Figure 2.6). If d is on the boundary of \mathbf{C} , then this extension does not cut any rectangles of $\mathcal{N}_{\mathbf{C}}$. Otherwise, we cut some rectangles. By the same reasoning as before, we need money (one dollar) only for the last rectangle that is cut. It is easy to see that in this case point d is in the interior of a maximal segment from $\mathcal{B}_{\mathbf{C}}$, say $\overline{g, h}$. By the same reasoning as applied before to points c , e , and f , we can conclude that the rectangle adjacent to segments $\overline{b, d}$ and $\overline{g, d}$ is not adjacent to a broken side of \mathbf{C} , so it had two dollars, and becomes adjacent to a broken side after the partition, so it provides us with that additional one dollar.

We now turn to the implementation of our algorithm. All the steps of our algorithm can be implemented in a manner similar to those in [dAF92], except the step that requires finding the segment $\overline{a, b}$. We describe the implementation assuming that the side $\overline{a, b}$ is horizontal and the unbroken side of \mathbf{C} is the right side of \mathbf{C} ; the other cases are similar. Note that we do not need to know or enumerate the rectangles whose sides contributed to the segment $\overline{a, b}$. Consider the set of horizontal boundary segments of all rectangles in our collection \mathbf{R} and join two segments if they share an endpoint. This will give us a set of segments in which each segment is a result of the joining of horizontal boundary segments of one or more rectangles. This can be easily obtained in $O(n \log n)$ time by sorting all the horizontal boundary segments of all rectangles in our collection \mathbf{R} by their distance from the y -axis. Finding the longest segment $\overline{a, b}$ and maintaining them for each partition that we generate can now be done by the same implementation as in [dAF92].

3. Lower bounds. In this section, we prove the following lower bounds.

THEOREM 3.1. *The following lower bounds hold:*

- (a) *There exists a configuration of n isothetic rectangles that form a tiling of the space such that any BSP tree for them is of size at least $\frac{3n}{2} - o(n)$.*
- (b) *If the rectangles are not required to form a tiling, there exists a configuration of n rectangles such that any BSP tree for them is of size at least $2n - o(n)$.*

Proof. We first focus on part (a) when the rectangles form a tiling.

We will construct the configuration in *stages*. See Figure 3.1. Stage 1 is the five rectangle configuration shown in the bottom left part of Figure 3.1. We construct stage 1 from stage 2 as follows. We reflect a stage 1 configuration along each axis as

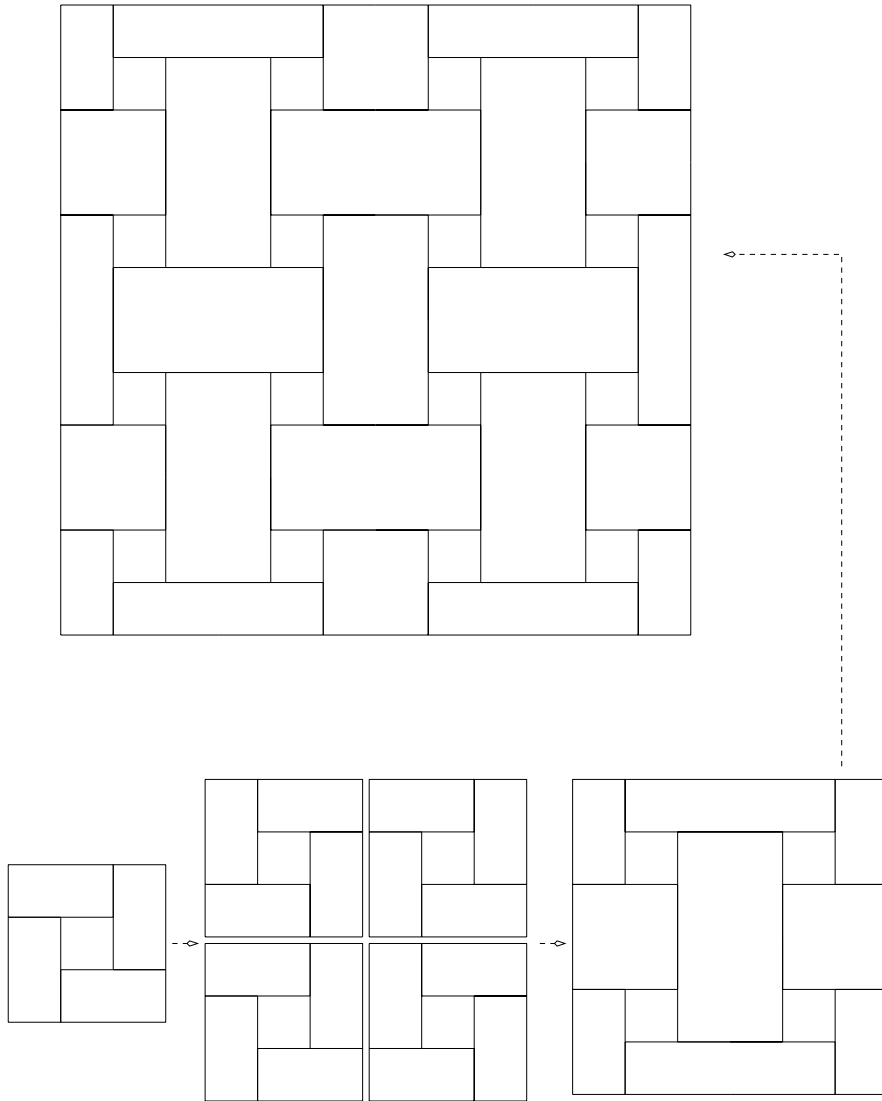
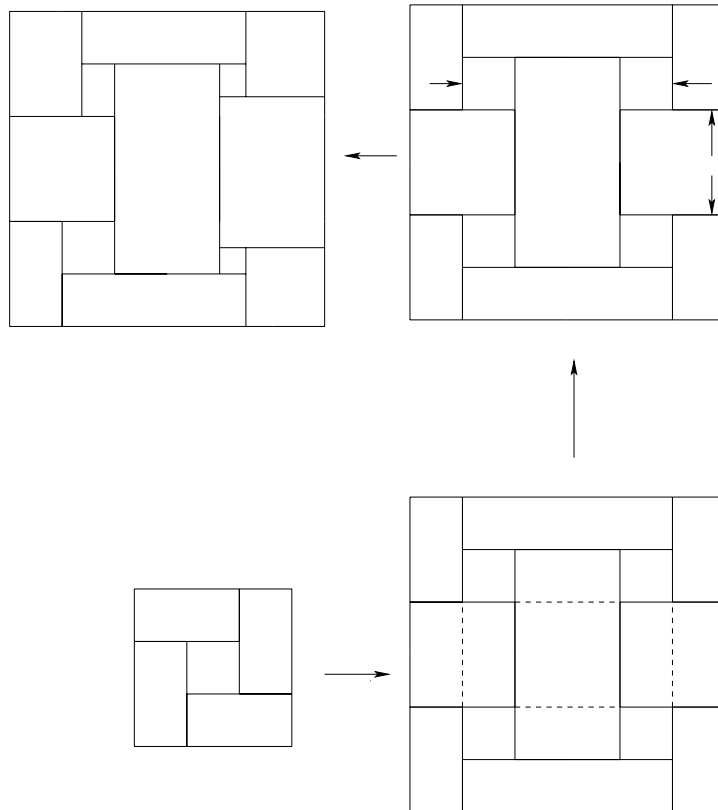


FIG. 3.1. Lower-bound construction for a set of isothetic rectangles that form a tiling. Construction of stages 1, 2, and 3.

well as across the lower right corner to get a configuration of four copies of stage 1. We merge any two mirror reflections abutting the borders of the four copies into one rectangle, whenever possible. The resulting configuration is shown in the bottom right part of Figure 3.1 with 13 rectangles. That is stage 2. Stage 3 consists of four copies of the configuration from stage 2 reflected and merged along the borders just as before, and so on.

Define B_i to be the number of rectangles on the boundary of any one side of a stage i configuration (from inside) and R_i to be total number of rectangles. For example, for Stage 1 $B_1 = 2$ and $R_1 = 5$ and for stage 2 $B_2 = 3$ and $R_2 = 13$.

FIG. 3.2. *Modifying stage 2.*

Furthermore, we have $B_i = 2B_{i-1} - 1$; this is because any boundary of Stage i consists of two copies of stage $i - 1$, but a single rectangle that abuts their joint is collapsed in the construction. By symmetry, all sides of stage i have B_i rectangles on the boundary. Also, note that

$$R_i = 4R_{i-1} - (4B_{i-1} - 1),$$

because in stage i we have the rectangles from four copies of stage $i - 1$, but on each of the four joints of the copies we lose one copy of the rectangles on the boundary and gain the one rectangle at the center. Solving the recurrences, we have $B_i = 2^i - 2^{i-1} + 1$, and $R_i = 2(4^{i-1}) + 2^i + 1$. The total number of rectangles in the final stage is n .

We will now modify the stages slightly, but it is critical.

Any BSP for stage 1 needs to cut one of the rectangles and therefore generates at least one additional rectangle. Our goal is to claim that any BSP for stage 2 needs at least four additional rectangles (for reasons that will be clear soon). The intuition is that each of the four copies of stage 1 that was used to generate stage 2 needs a separate BSP cut and therefore will generate a separate additional rectangle in the BSP. This is not true as such for the 13 rectangle configuration shown in Figure 3.1 and reproduced as the bottom-right configuration in Figure 3.2. As the two parallel

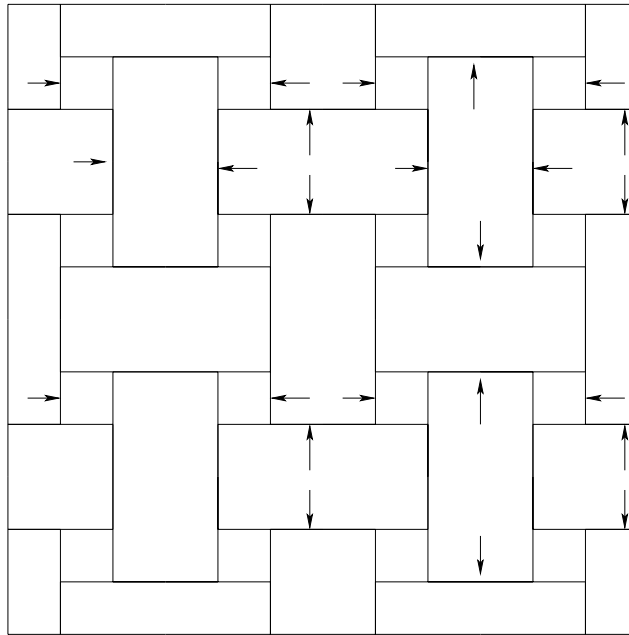


FIG. 3.3. *Modifying stage 3. Solid arrows indicate in what direction the side of a rectangle has to be moved slightly.*

sets of dotted lines show, a single cut can serve as the BSP cut of two copies juxtaposed horizontally or vertically. This is because of the symmetry of our construction. In order to break this symmetry, we need to make sure that the vertical line of one of the top two copies of stage 1 does not align with the vertical line in the copy of stage 1 directly underneath it. This is accomplished by moving the vertical lines marked in the figure a little along the direction of the arrow. The process is similar for the horizontal lines of the right two copies of stage 1. This results in the modified stage 2 shown in Figure 3.2. Now we need at least four additional rectangles in any BSP tree of Stage 2.

Modifying stage 3 is similar. Refer to Figure 3.3. We start with the modified stage 2 and repeat the construction of stage 3 as before. Again, we need to make sure that the horizontal lines in the top (bottom) left copy do not line up with those in the top (bottom) right copy and the vertical lines in the top left (right) copy do not line up with the bottom left (right) copy. This is accomplished by moving various line segments a little *further* along the direction shown in the figure. In this manner, we can ensure that the individual BSP cuts of each of the four copies of stage 2 are all needed in the BSP of stage 3. As a result, we have $4 * 4 = 16$ additional rectangles for the BSP of modified stage 3. It is easy to see that we can repeat this procedure to construct modified stage i such that any BSP of it will need 4^i additional rectangles.

That completes the description of the configuration. Clearly, B_i 's and R_i 's remain unchanged during the modification of stages.

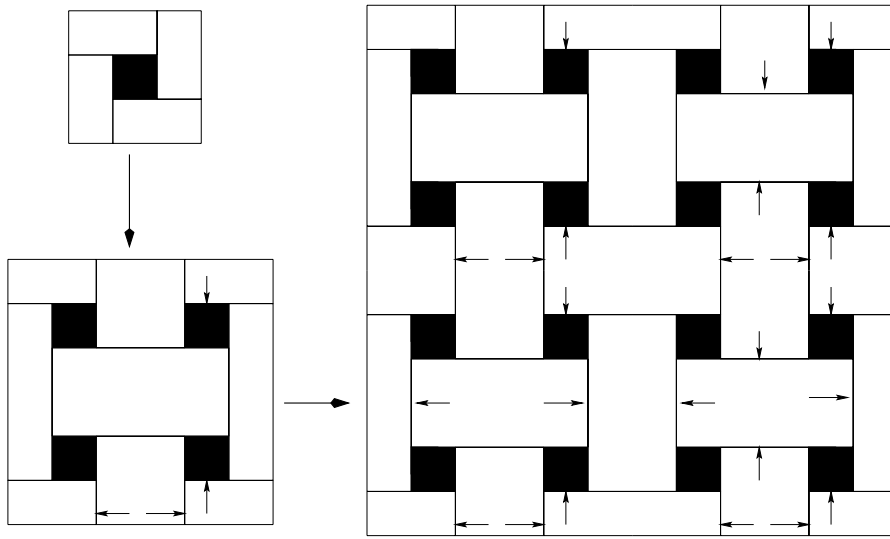


FIG. 3.4. Lower bound construction for a set of isothetic rectangles, not necessarily forming a tiling. The black region is a hole.

We will now complete the lower bound calculation. Define C_i to be the number of additional rectangles (in excess of R_i) needed in any binary space partition for modified stage i . From our construction, $C_i = 4C_{i-1}$; solving the recurrence, we get $C_i = 4^{i-1}$. Define the lower bound factor on the BSP tree size for a given set of rectangles to be the ratio of the size of a *minimum-size* BSP tree to the given number of rectangles. Clearly, for our construction, the lower bound factor on the BSP tree size is $\frac{C_i + R_i}{R_i} = 1 + \frac{C_i}{R_i}$. From the formulas for C_i and R_i , we have that the lower bound factor on the size of the BSP tree is at least

$$1 + \frac{4^{i-1}}{2(4^{i-1}) + 2^i + 1} = \frac{3}{2} - \frac{2^i + 1}{2(4^{i-1}) + 2^i + 1} \geq \frac{3}{2} - \frac{\sqrt{2n} + 1}{n} = \frac{3}{2} - o(1).$$

For the case when the rectangles are not required to be a tiling, we repeat the same construction as above, except that we leave the central square as a hole. The first two stages are shown in Figure 3.4, where the shaded region is a hole. Repeating the argument above (with $n = 4^{i-1} + 2^i + 1$), we obtain the lower bound factor to be at least

$$1 + \frac{4^{i-1}}{4^{i-1} + 2^i + 1} = 2 - \frac{2^i + 1}{4^{i-1} + 2^i + 1} \geq 2 - \frac{2\sqrt{n} + 1}{n} = 2 - o(1). \quad \square$$

4. Improved rectangle tiling algorithms. We present improved approximation algorithms for several dual rectangle tiling problems. These rectangular tiling problems and our improved approximation algorithms for them are of interest in many areas: databases, parallel load balancing, spatial data structures, video compression, etc. As an example, consider the following problem [MPS99].

Dual rectangle tiling (DR TILE) problem. Given an $n \times n$ array A of positive integers and a parameter $\delta > 0$, the problem is to tile (partition) A with the minimum number of axis-parallel rectangular tiles such that the maximum weight of any tile is at most δ . The *weight* of a tile is the sum squared error of each entry from the average of the entries in that tile.

The DRTILE problem is known to be NP-hard [MPS99]. Our new results are as follows. The previously best known algorithm for (a) in the following theorem also used no more than twice as many tiles as needed, but its running time was $O(n^{\geq 10})$ [KMP98]! The previously best known algorithm for (b) in the following theorem with the same running time used $4p^*O(1/\epsilon^2)$ tiles⁷ and hence was *worse* by a factor of 2 in approximation ratio [MPS99].

THEOREM 4.1. *The following results hold for the above DRTILE problem:*

- (a) *There is an $O(n^5)$ time algorithm to find a tiling with maximum weight of a tile being at most δ using no more than twice as many tiles as needed.*
- (b) *There is an $O(n^{2+\epsilon})$ time algorithm that uses at most $2p^*O(1/\epsilon^2)$ tiles, where p^* is the minimum number of tiles needed and $\epsilon > 0$ is arbitrarily small.*

Proof. Proofs of both (a) and (b) depend on the following argument. Assume that the optimum tiling has p^* tiles. Theorem 2.2 implies that there is a BSP tree for those tiles of size at most $2p^*$. Since the weight of a subtile of a tile is no more than that of the tile itself,⁸ this means that there exists a BSP of the tiling with at most $2p^*$ tiles in which the weight of each tile at most δ . We can find a BSP of *minimum* size among BSPs of all possible tiling of the array A in which the weight of each tile at most δ in $O(n^5)$ time by dynamic programming [KMP98, MPS99]. This approximates p^* by a factor of 2 and proves (a). For part (b), we use the same reasoning as before but apply the sparse and rounded dynamic programming techniques in [MPS99] to reduce the running time at the expense of increasing the size of the computed BSP. \square

Remark 1. Our lower bound result in Theorem 3.1(a) show that alternate approach is needed in order to get significantly better approximations for this problem.

Remark 2. The improvements described in Theorem 4.1 also hold for other tiling problems such as with either the maximum or the sum of weights of tiles, different weight functions, etc. The claims of Theorem 4.1 also apply to the special case (the DRTILE problem considered in [BDMR01, KMP98] and elsewhere) when the weight of a tile is the sum of all array elements in it. Many results exist for the DRTILE problem which use the special properties of the weight function and obtain better bounds than the ones we have quoted above. However, the strength of our results here is that they apply to many more complex weight functions, sum-squared-error being one example. The technical condition for the results in Theorem 4.1 to hold is that the weight functions have to be *superadditive*. Informally, this means that splitting the tiles in any optimum solution does not make it worse in the criteria. (A formal definition can be found in [MPS99].)

5. Concluding remarks. We have shown upper and lower bounds on the size of a BSP tree for a set of n isothetic rectangles. In addition, our results give improved approximation algorithms for rectangular tiling problems that arise in many application areas [BDMR01, MPS99]. We leave open the problem of closing the gap between the upper and lower bounds and of developing such detailed analyses for BSP trees for other objects.

⁷For specific values of ϵ , e.g., $\epsilon = \frac{3}{4}$, one can calculate the approximation ratio as well as the running time from [MPS99], and it is quite reasonable. However, for arbitrarily small but fixed ϵ , the bound appears to be substantially large. Hence, we retain just the expression $O(1/\epsilon^2)$ in the description.

⁸This is because the weight function is superadditive; see also the end of Remark 2.

Acknowledgments. We would like to thank the anonymous reviewers for helpful suggestions, which led to a substantially improved presentation of the results reported in this paper, and Prof. Ravi Janardan for help in the $O(n \log n)$ time implementation of the algorithm in Theorem 2.1.

REFERENCES

- [AG+96] P. AGARWAL, E. GROVE, T. MURALI, AND J. VITTER, *Binary space partitions for fat rectangles*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 482–491.
- [AS94] P. AGARWAL AND S. SURI, *Surface approximation and geometric partitions*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 1994, pp. 24–33.
- [B93] C. BALLIEUX, *Motion Planning Using Binary Space Partitions*, Technical report, Utrecht University, Utrecht, The Netherlands, 1993.
- [BDMR01] P. BERMAN, B. DASGUPTA, S. MUTHUKRISHNAN, AND S. RAMASWAMI, *Improved approximation algorithms for rectangle tiling and packing*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 2001, pp. 427–436.
- [CF89] S. CHIN AND S. FEINER, *Near real-time shadow generation using BSP trees*, Comput. Graphics, 23 (1989), pp. 99–106.
- [dAF92] F. D’AMORE AND P. G. FRANCIOSA, *On the optimal binary plane partition for sets of isothetic rectangles*, Inform. Process. Lett., 44 (1992), pp. 255–259.
- [BW80] J. L. BENTLEY AND D. WOOD, *An optimal worst-case algorithm for reporting intersections of rectangles*, IEEE Trans. Comput., 29 (1980), pp. 571–577.
- [deGO] M. DE BERG, M. DE GROOT, AND M. OVERMARS, *New results on binary space partitions in the plane*, in Proceedings of the 4th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 824, Springer-Verlag, Berlin, 1994, pp. 61–72.
- [DMM01] A. DUMITRESCU, J. MITCHELL, AND M. SHARIR, *Binary space partitions for axis-parallel segments, rectangles and hyperrectangles*, in Proceedings of the 17th ACM Symposium on Computational Geometry, 2001, pp. 141–150.
- [D94] S. E. DORWARD, *A survey of object-space hidden surface removal*, Internat. J. Comput. Geom. Appl., 4 (1994), pp. 325–362.
- [FKN80] H. FUCHS, Z. KEDEM, AND B. NAYLOR, *On visible surface generation by a priori tree structures*, Comput. Graph., 14 (1980), pp. 124–133.
- [NW95] V. HAI NGUYEN AND P. WIDMAYER, *Binary Space Partitions for Sets of Hyperrectangles*, Lecture Notes in Comput. Sci. 1023, Springer-Verlag, Berlin, 1995.
- [KMP98] S. KHANNA, S. MUTHUKRISHNAN, AND M. PATERSON, *Approximating rectangle tiling and packing*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 1998, pp. 384–393.
- [M85] E. M. MCCREIGHT, *Priority search trees*, SIAM J. Comput., 14 (1985), pp. 257–276.
- [M90] J. MITCHELL, *On maximum flows in polyhedral domains*, J. Comput. System Sci., 40 (1990), pp. 88–123.
- [MPS99] S. MUTHUKRISHNAN, V. POOSALA, AND T. SUEL, *Rectangular partitionings: Algorithms, complexity and applications*, in Proceedings of the Seventh International Conference on Database Theory, Lecture Notes in Comput. Sci. 1540, Springer-Verlag, Berlin, 1999, pp. 236–256.
- [NT86] B. NAYLOR AND W. THIBAUT, *Application of BSP Trees to Ray-Tracing and CSG Evaluation*, Technical report GIT-ICS 86/03, Georgia Tech, Atlanta, GA, 1986.
- [PY90] M. PATERSON AND F. YAO, *Efficient binary space partitions for hidden-surface removal and solid modeling*, Discrete Comput. Geom., 5 (1990), pp. 485–503.
- [PY92] M. PATERSON AND F. YAO, *Optimal binary space partitions for orthogonal objects*, J. Algorithms, 13 (1992), pp. 99–113.
- [S90] H. SAMET, *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*, Addison-Wesley, Reading, MA, 1990.
- [T92] S. TELLER, *Visibility Computations in Densely Occluded Polyhedral Environments*, Ph.D. Thesis, Department of Computer Science, University of California, Berkeley, Berkeley, CA, 1992.

- [TN87] W. THIBAUT AND B. NAYLOR, *Set operations on polyhedra using binary space partitioning trees*, *Comput. Graphics*, 21 (1987), pp. 153–162.
- [T01] C. D. TÓTH, *A note on binary plane partitions*, in *Proceedings of the 17th ACM Symposium on Computational Geometry*, 2001, pp. 151–156.
- [van90] P. VAN OOSTEROM, *A modified binary space partition for geographic information systems*, *Int. J. GIS*, 4 (1990), pp. 133–146.

GRAPHS WITH CONNECTED MEDIANS*

HANS-JÜRGEN BANDEL† AND VICTOR CHEPOI‡

Abstract. The median set of a graph G with weighted vertices comprises the vertices minimizing the average weighted distance to the vertices of G . We characterize the graphs in which, with respect to any nonnegative vertex weights, median sets always induce connected subgraphs. The characteristic conditions can be tested in polynomial time (by employing linear programming) and are immediately verified for a number of specific graph classes.

Key words. graphs, medians, local medians, majority rule, LP duality

AMS subject classifications. 06A12, 90A08

PII. S089548019936360X

1. Introduction. Given a (finite, connected) graph G one is sometimes interested in finding the vertices minimizing the total distance

$$F(x) = \sum_u d(u, x)$$

to the vertices u of G , where the distance $d(u, x)$ between u and x is the length of a shortest path connecting u and x . The subgraph induced by all minima of F need not be connected: actually every (possibly disconnected) graph can be realized as such a “median” subgraph of another graph (Slater [26]); see Hendry [20] for further information and pertinent references. The median subgraph of an unweighted graph is also known as the “distance center” [25]. It is also used to derive fixed subgraph theorems [9] and other results [19]. Here we will focus on the weighted version of the median problem, which arises with one of the basic models in discrete facility location [28, 29] and with majority consensus in classification and data analysis [3, 10, 11, 12, 24]. A *weight function* is any mapping π from the vertex set to the nonnegative real numbers, which is not the zero function (in order to avoid trivialities). The total weighted distance of a vertex x in G is given by

$$F_\pi(x) = \sum_u \pi(u)d(u, x).$$

A vertex x minimizing this expression is a *median* (vertex) of G with respect to π , and the set of all medians is the *median set* $\text{Med}(\pi)$. By a *local median* one means a vertex x such that $F_\pi(x)$ does not exceed $F_\pi(y)$ for any neighbor y of x . Denote by $\text{Med}_{\text{loc}}(\pi)$ the set of all local medians. We will consider the following questions:

- When are all median sets $\text{Med}(\pi)$ of a graph connected?
- When does $\text{Med}(\pi) = \text{Med}_{\text{loc}}(\pi)$ hold for all weight functions π ?

*Received by the editors November 3, 1999; accepted for publication (in revised form) March 12, 2002; published electronically April 8, 2002.

<http://www.siam.org/journals/sidma/15-2/36360.html>

†Fachbereich Mathematik, Universität Hamburg, Bundesstr. 55, D-20146 Hamburg, Germany (bandelt@math.uni-hamburg.de).

‡Laboratoire d’Informatique Fondamentale, Université de la Méditerranée, Faculté des Sciences de Luminy, F-13288 Marseille Cedex 9, France (victor.chepoi@lim.univ-mrs.fr). The research of this author was supported in part by the Alexander von Humboldt Stiftung.

If we allow only 0-1 weight functions π , then recognition of graphs with $\text{Med}(\pi) = \text{Med}_{\text{loc}}(\pi)$ is an *NP*-complete problem [2]. We will show that connectivity of median sets with respect to arbitrary weight functions turns out to be equivalent to the condition that for each function F_π all local minima are global. This property can also be formulated as a certain convexity condition.

In the next section we investigate weakly convex functions on graphs, and in section 3 we then obtain the basic characterizations of graphs with connected medians. One of those equivalent conditions, weak convexity of F_π for each weight function π , can be formulated as a linear programming problem, thus allowing us to recognize graphs with connected medians in polynomial time. This LP condition, however, entails a lot of redundancy and cannot be read off from other graph properties, which are known to imply median connectedness. To establish the main result of section 4, we split the LP condition into a rather trivial part (requiring that metric triangles be equilateral) and a local condition (involving the intervals between vertices at distance 2) to which LP duality is applied. This theorem can conveniently be used to derive median properties in several specific classes of graphs, as is demonstrated in the final section.

2. Weakly convex functions. In what follows all graphs are assumed to be finite and connected. A real-valued function f defined on the vertex set V of a graph G is said to be *weakly convex* if for any two vertices u, v , and a real number λ between 0 and 1 such that $\lambda d(u, v)$ and $(1 - \lambda)d(u, v)$ are integers there exists a vertex x such that

$$d(u, x) = \lambda d(u, v), \quad d(v, x) = (1 - \lambda)d(u, v),$$

$$f(x) \leq (1 - \lambda)f(u) + \lambda f(v).$$

Weakly convex functions were introduced by Arkhipova and Sergienko [1] under the name “ r -convex functions”; see also Lebedeva, Sergienko, and Soltan [21].

The *interval* $I(u, v)$ between two vertices u and v consists of all vertices on shortest u, v -paths; that is,

$$I(u, v) = \{x : d(u, x) + d(x, v) = d(u, v)\}.$$

For convenience we will use the shorthand

$$I^\circ(u, v) = I(u, v) - \{u, v\}$$

to denote the “interior” of the interval between u and v .

LEMMA 2.1. *For a real-valued function f defined on the vertex set of a graph G the following conditions are equivalent:*

- (i) f is weakly convex;
- (ii) for any two nonadjacent vertices u and v there exists $w \in I^\circ(u, v)$ such that

$$d(u, v) \cdot f(w) \leq d(v, w) \cdot f(u) + d(u, w) \cdot f(v);$$

- (iii) any two vertices u and v at distance 2 have a common neighbor w with

$$2f(w) \leq f(u) + f(v).$$

Proof. (i) \Rightarrow (ii) \Rightarrow (iii) is trivial.

(iii) \Rightarrow (i): Consider two vertices u and v at distance $d(u, v) = n$. Among all shortest paths connecting u and v select a path $P = (u = w_0, w_1, \dots, w_{n-1}, w_n = v)$ such that $\sum_{i=0}^n f(w_i)$ is as small as possible. Condition (iii) implies that $2f(w_i) \leq f(w_{i-1}) + f(w_{i+1})$ for each $1 < i < n$. Regard

$$(0, f(w_0)), (1, f(w_1)), \dots, (n-1, f(w_{n-1})), (n, f(w_n))$$

as points in the plane \mathbb{R}^2 . Connecting the consecutive points by segments, we will get a graph of a piecewise-linear function. This function is necessarily convex (in the usual sense), because it coincides on P with the function f . From this we conclude that

$$f(w_i) \leq \lambda f(u) + (1 - \lambda)f(v),$$

where $\lambda = (n - i)/n$ and $1 - \lambda = i/n$. \square

Recall that a real-valued function f defined on a path $P = (w_0, w_1, \dots, w_p)$ is *peakless* [14, p. 109] if $0 \leq i < j < k \leq p$ implies $f(w_j) \leq \max\{f(w_i), f(w_k)\}$ and equality holds only if $f(w_i) = f(w_k)$. Now, we will say that a function f defined on the vertex set of a graph G is *pseudopeakless* if any two vertices of G can be joined by a shortest path along which f is peakless [16]. The function f is called *unimodal* if every local minimum of f is global; that is, if the inequality $f(v) \leq f(y)$ holds for all neighbors y of a vertex v of G , then it holds for all vertices y of G . The next result shows that weakly convex functions are pseudopeakless and pseudopeakless functions are unimodal.

Remark 1. The following conditions are equivalent for every real-valued function f defined on the vertex set of a graph G :

- (i) f is pseudopeakless;
- (ii) for any two nonadjacent vertices u, v there is a vertex $w \in I^\circ(u, v)$ such that $f(w) \leq \max\{f(u), f(v)\}$ and equality holds only if $f(u) = f(v)$;
- (iii) the composition $\alpha \circ f$ is weakly convex for some strictly isotone transformation α of the reals (i.e., $\alpha(r) < \alpha(s)$ for any two reals $r < s$).

If one of the conditions (i)–(iii) is satisfied, then the function f is unimodal.

Proof. (i) \Leftrightarrow (ii) is trivial. The property of being pseudopeakless is invariant under strictly isotone transformations of the range, whence (iii) \Rightarrow (i).

(i) \Rightarrow (iii): Let f be a pseudopeakless function taking n distinct values $a_1 < a_2 < \dots < a_n$. Let α be a strictly isotone map which assigns to each a_i the integer 2^i . We assert that the composition $\alpha \circ f$ is weakly convex. For given vertices u and v at distance 2, let w be their common neighbor such that f is peakless along the path (u, w, v) . Say $a_i = f(u) \leq f(v) = a_k$ and $a_j = f(w) \leq a_k$. If $i = k$, then $i = j = k$, and hence $\alpha \circ f(w) = \alpha \circ f(u) = \alpha \circ f(v)$. Else we have $i < k$ and thus $j < k$, and consequently

$$2\alpha \circ f(w) = 2^{j+1} \leq 2^k < 2^i + 2^k = \alpha \circ f(u) + \alpha \circ f(v),$$

as required.

To establish unimodality of a pseudopeakless function f , let u be a global minimum and v a local minimum of f . Consider a shortest path P between u and v along which f is peakless. Then for any neighbor w of v we have $f(w) \geq \max\{f(u), f(v)\}$, whence $f(u) = f(v) = f(w)$, as required. \square

In what follows we will apply Lemma 2.1 to the functions F_π of G . The simplest instance is given by the weight function π assigning 1 to a distinguished vertex w and

0 otherwise: then $F_\pi = d(\cdot, w)$ just measures the distance in G to that vertex. We say that G is *meshed* [6] if the function $d(\cdot, w)$ is weakly convex for every choice of w . A somewhat weaker property is used in subsequent results. Three vertices u, v, w of a G are said to form a *metric triangle* uvw if the intervals $I(u, v)$, $I(v, w)$, and $I(w, u)$ pairwise intersect only in the common end vertices. If $d(u, v) = d(v, w) = d(w, u) = k$, then this metric triangle is called *equilateral* of *size* k .

Remark 2. Every metric triangle in a meshed graph G is equilateral.

Proof. To see this, suppose the contrary: let uvw be a metric triangle in G with $d(u, w) < d(v, w)$. For $\lambda = 1 - 1/d(v, w)$, weak convexity of $f = d(\cdot, w)$ provides us with a vertex x such that

$$\begin{aligned} d(u, x) &= d(u, v) - 1, d(v, x) = 1, \\ d(w, x) &\leq d(u, w)/d(u, v) + d(v, w)(1 - 1/d(u, v)) \\ &< d(v, w)/d(u, v) + d(v, w)(1 - 1/d(u, v)) = d(v, w). \end{aligned}$$

This implies $x \in I(u, v) \cap I(v, w)$, a contradiction. \square

3. Basic characterizations. We commence by giving first answers to the questions raised in the introduction. A connected subgraph H of a graph G is *isometric* if the distance function d of G restricts to the distance function of H .

PROPOSITION 3.1. *For a graph G the following conditions are equivalent:*

- (i) $Med_{loc}(\pi) = Med(\pi)$ for all weight functions π ;
- (ii) F_π is weakly convex for all π ;
- (iii) F_π is pseudopeakless for all π ;
- (iv) all level sets $\{x : F_\pi(x) \leq \lambda\}$ induce isometric subgraphs;
- (v) all median sets $Med(\pi)$ induce isometric subgraphs;
- (vi) all median sets $Med(\pi)$ are connected.

Any of the conditions (i)–(vi) is equivalent to the analogous condition with the additional requirement that π be positive.

The following observation is basic for the proof of Proposition 3.1.

LEMMA 3.2. *If the function F_π is not weakly convex on the vertex set V of a graph G for some weight function π , then there exist a positive weight function π^+ and vertices u, v at distance 2 such that $Med(\pi^+) = \{u, v\}$.*

Proof. If F_π is not weakly convex, then by Lemma 2.1 there exist two vertices u and v at distance 2 such that $2F_\pi(w) > F_\pi(u) + F_\pi(v)$ for all (common neighbors) $w \in I^\circ(u, v)$. This inequality can be maintained under sufficiently small positive perturbations of π : viz., add any δ satisfying

$$0 < \delta \cdot 2 \cdot \#V < \min_{w \in I^\circ(u, v)} (2F_\pi(w) - F_\pi(u) - F_\pi(v))$$

to all weights, yielding the new weights $\pi'(x) = \pi(x) + \delta$. Then

$$\begin{aligned} &2F_{\pi'}(w) - F_{\pi'}(u) - F_{\pi'}(v) \\ &= 2F_\pi(w) - F_\pi(u) - F_\pi(v) - \delta \cdot \sum_{x \in V} (d(u, x) + d(v, x) - 2d(w, x)) \\ &\geq 2F_\pi(w) - F_\pi(u) - F_\pi(v) - \delta \cdot 2 \cdot \#V > 0 \end{aligned}$$

for all $w \in I^\circ(u, v)$; that is, the initial inequality remains valid with respect to the thus perturbed weight function. We may therefore assume that π is actually positive.

We stipulate that $\epsilon = \frac{1}{2}(F_\pi(v) - F_\pi(u)) \geq 0$. Let μ denote the maximum value of F_π . Define a new positive weight function π^+ by

$$\pi^+(u) = \pi(u) + 2\mu, \quad \pi^+(v) = \pi(v) + 2\mu + \epsilon,$$

and $\pi^+(x) = \pi(x)$ otherwise. Then

$$F_{\pi^+}(u) = F_\pi(u) + 2(2\mu + \epsilon) = F_\pi(v) + 4\mu = F_{\pi^+}(v)$$

and $\text{Med}(\pi^+) \subseteq I(u, v)$ because

$$F_{\pi^+}(x) \geq F_\pi(x) + 2\mu(d(u, x) + d(v, x)) \geq F_\pi(x) + 6\mu > F_{\pi^+}(u)$$

for every vertex x outside $I(u, v)$. For $w \in I^\circ(u, v)$ one obtains

$$F_{\pi^+}(w) = F_\pi(w) + 4\mu + \epsilon > \frac{1}{2}(F_\pi(u) + F_\pi(v)) + 4\mu + \epsilon = F_{\pi^+}(u).$$

Therefore $\text{Med}(\pi^+)$ consists only of u and v . This concludes the proof. \square

Proof of Proposition 3.1. The implications (ii) \Rightarrow (iii) \Rightarrow (iv) \Rightarrow (v) \Rightarrow (vi) and (ii) \Rightarrow (i) are trivial, while (vi) \Rightarrow (ii) is covered by Lemma 3.2. It remains to verify that (i) implies (ii). Suppose by way of contradiction that some function F_π is not weakly convex. By Lemma 3.2 there exist a positive weight function π^+ and vertices u, v at distance 2 such that $\text{Med}(\pi^+) = \{u, v\}$. Pick any ϵ satisfying

$$0 < \epsilon < \min_{x \in V - \{u, v\}} F_{\pi^+}(x) - F_{\pi^+}(v)$$

and define a new weight function π' by

$$\pi'(u) = \pi^+(u) + \epsilon$$

and $\pi'(x) = \pi^+(x)$ otherwise. Then

$$F_{\pi'}(u) = F_{\pi^+}(u) = F_{\pi^+}(v),$$

$$F_{\pi'}(x) = F_{\pi^+}(x) + d(x, u)\epsilon \geq F_{\pi^+}(x) + \epsilon > F_{\pi^+}(v) + 2\epsilon = F_{\pi'}(v)$$

for all $x \in V - \{u, v\}$. Therefore both u and v are local minima of $F_{\pi'}$, but $v \notin \text{Med}(\pi')$. This establishes the implication (i) \Rightarrow (ii).

The same arguments can be applied to prove that the analogous conditions (i⁺)–(vi⁺) additionally requiring the weight functions to be positive are all equivalent. Since (i) \Rightarrow (i⁺) is trivial and (vi⁺) \Rightarrow (i) is covered by Lemma 3.2, the proof is complete. \square

In view of Lemma 2.1(iii) and Proposition 3.1, all median sets are connected if and only if for each pair u, v of vertices at distance 2 the following system of linear inequalities is unsolvable in π :

$$D^{uv}\pi < 0 \text{ and } \pi \geq 0 \text{ with matrix}$$

$$D^{uv} = (d(u, x) + d(v, x) - 2d(w, x))_{w \in I^\circ(u, v), x \in V}.$$

Since LP problems can be solved in polynomial time, we thus obtain the following result.

COROLLARY 3.3. *The problem to decide whether all median sets $\text{Med}(\pi)$ of a graph G are connected is solvable in polynomial time.*

4. The main result. In order to obtain a convenient characterization of median connectedness, we will restrict the supports of the weight functions π to be tested and then take advantage of LP duality in the following form.

Remark 3. Let u, v be any vertices of a graph G with $d(u, v) = 2$, and let W be a nonempty subset of $I^\circ(u, v)$ and X be a nonempty subset of the vertex set of G . Then for every weight function π with support included in X there exists some $w \in W$ such that

$$F_\pi(u) + F_\pi(v) - 2F_\pi(w) \geq 0$$

if and only if there exists a weight function η with support included in W such that for every $x \in X$

$$\sum_{w \in W} (d(u, x) + d(v, x) - 2d(w, x))\eta(w) \geq 0.$$

Proof. Let D denote the submatrix of D^{uv} (as just defined) with rows and columns corresponding to W and X , respectively. The asserted equivalence is then a particular instance of LP duality, as formulated by J.A.Ville (cf. [17, p. 248]) in terms of systems of linear inequalities:

$$D\pi < 0, \pi \geq 0 \text{ is unsolvable} \iff \eta D \geq 0, \eta \geq 0, \eta \neq 0 \text{ is solvable.} \quad \square$$

For vertices x and y define the following superset of $I(x, y)$:

$$J(x, y) = \{z \in V : I(z, x) \cap I(z, y) = \{z\}\}.$$

PROPOSITION 4.1. *The median sets $Med(\pi)$ in a graph G are connected for all weight functions π if and only if (1) all metric triangles are equilateral and (2) for any vertices u, v with $d(u, v) = 2$ and every weight function π whose support $\{x \in V : \pi(x) > 0\}$ is included in the set $J(u, v)$ there exists a vertex $w \in I^\circ(u, v)$ such that $2F_\pi(w) \leq F_\pi(u) + F_\pi(v)$.*

Proof. Necessity immediately follows from Remark 2 and Proposition 3.1. To prove sufficiency assume that (1) and (2) hold, but there exists a weight function π for which F_π is not weakly convex. Then by Lemma 2.1 there are vertices u and v at distance 2 such that

$$2F_\pi(w) > F_\pi(u) + F_\pi(v) \quad \text{for all } w \in I^\circ(u, v).$$

Now, fixing u and v , we may assume that π was chosen so that this inequality still holds and the number of vertices from the support of π outside $J(u, v)$ is as small as possible. By (2) we can find a vertex x with $\pi(x) > 0$ and $x \notin J(u, v)$. To arrive at a contradiction, we will slightly modify π by setting the weight of x to zero (and possibly transferring its old weight to some vertex in $J(u, v)$). We distinguish three cases.

Case 1. $|d(u, x) - d(v, x)| = 2$.

Then $2d(w, x) = d(u, x) + d(v, x)$ for every $w \in I^\circ(u, v)$. Hence the weight function π' defined by

$$\pi'(y) = \begin{cases} \pi(y) & \text{if } y \neq x, \\ 0 & \text{if } y = x \end{cases}$$

satisfies $2F_{\pi'}(w) > F_{\pi'}(u) + F_{\pi'}(v)$, contrary to the minimality assumption on π .

Case 2. $|d(u, x) - d(v, x)| = 1$.

Without loss of generality assume that $d(v, x) = d(u, x) + 1$. Let x' be a vertex in $I(u, x) \cap I(v, x)$ at maximal distance to x . Then $x' \in J(u, v)$ and $d(v, x') = d(u, x') + 1$. From the latter equality and $d(u, v) = 2$ we infer that $I(u, v) \cap I(u, x') = \{u\}$. Let t be a vertex in $I(u, v) \cap I(v, x')$ at maximal distance to v . Since $u \notin I(v, x)$, the vertices u, t, x' must constitute an equilateral metric triangle. Therefore $d(u, x') = d(t, u) = 1$ and $d(v, x') = 2$. Define a new weight function π' , where

$$\pi'(y) = \begin{cases} \pi(y) & \text{if } y \neq x, x', \\ 0 & \text{if } y = x, \\ \pi(x) + \pi(x') & \text{if } y = x'. \end{cases}$$

Let $k = d(u, x) = d(v, x) - 1$. Then

$$F_{\pi'}(u) + F_{\pi'}(v) = F_{\pi}(u) + F_{\pi}(v) - (2k - 2)\pi(x).$$

For every $w \in I^\circ(u, v)$ we have either

$$d(w, x) = k \quad \text{and} \quad 1 \leq d(w, x') \leq 2$$

or

$$d(w, x) = k + 1 \quad \text{and} \quad d(w, x') = 2.$$

Hence

$$F_{\pi'}(w) = F_{\pi}(w) - (d(w, x) - d(w, x'))\pi(x) \geq F_{\pi}(w) - (k - 1)\pi(x),$$

and further

$$\begin{aligned} & 2F_{\pi'}(w) - F_{\pi'}(u) - F_{\pi'}(v) \\ & \geq 2F_{\pi}(w) - 2(k - 1)\pi(x) - F_{\pi}(u) - F_{\pi}(v) + 2(k - 1)\pi(x) \\ & = 2F_{\pi}(w) - F_{\pi}(u) - F_{\pi}(v) > 0, \end{aligned}$$

again a contradiction to the choice of π .

Case 3. $d(u, x) = d(v, x)$.

Let x' be a vertex in $I(u, x) \cap I(v, x)$ at maximal distance to x . Since all metric triangles of G are equilateral and $d(u, v) = 2$, we obtain that

$$1 \leq d(x', u) = d(x', v) \leq 2.$$

Define the new weight function π' as in Case 2. Then

$$F_{\pi'}(u) = F_{\pi}(u) - d(x, x')\pi(x),$$

$$F_{\pi'}(v) = F_{\pi}(v) - d(x, x')\pi(x),$$

and for every $w \in I^\circ(u, v)$

$$\begin{aligned} F_{\pi'}(w) &= F_\pi(w) - (d(x, w) - d(w, x'))\pi(x) \\ &\geq F_\pi(w) - d(x, x')\pi(x), \end{aligned}$$

a final contradiction. \square

In view of Remark 2 we may require in Proposition 4.1 (as well as Theorem 4.4 below) that G be meshed instead and that all metric triangles in G be equilateral.

Let $M(u, v)$ denote the set of those vertices of $J(u, v)$ which are equidistant to u and v :

$$M(u, v) = \{x \in V : I(x, u) \cap I(x, v) = \{x\} \text{ and } d(u, x) = d(v, x)\}.$$

The neighborhood $N(x)$ of a vertex x consists of all vertices adjacent to x . Note that if $d(u, v) = 2$, then $N(u) \cap N(v) = I^\circ(u, v) \subseteq M(u, v)$.

LEMMA 4.2. *If all median sets of a graph G are connected, then for any vertices u and v at distance 2 there exist (not necessarily distinct) vertices $s, t \in I^\circ(u, v)$ such that*

$$d(s, y) + d(t, y) \leq d(u, y) + d(v, y)$$

for all vertices $y \in M(u, v)$.

Proof. We define a weight function π for which the hypothesis that F_π be pseudopeakless implies the asserted inequality: let

$$\pi(x) = \begin{cases} 0 & \text{if } x \notin M(u, v), \\ 4 & \text{if } x \in M(u, v) - I^\circ(u, v), \\ 2 & \text{if } x \in I^\circ(u, v) \text{ with } \#(I^\circ(u, v) \cap N(x)) \geq \#I^\circ(u, v) - 2 \\ & \text{and } d(x, y) = 2 \text{ for all } y \in M(u, v) - I^\circ(u, v), \\ 3 & \text{otherwise.} \end{cases}$$

Since weights are zero outside $M(u, v)$, we have $F_\pi(u) = F_\pi(v)$. Select $s \in I^\circ(u, v)$ such that F_π is peakless along the (shortest) path (u, s, v) , that is, $F_\pi(s) \leq F_\pi(u) = F_\pi(v)$. Note that

$$d(u, x) + 1 \geq d(s, x) \geq d(u, x) = d(v, x) \quad \text{for all } x \in M(u, v) - \{s\}.$$

We claim that $\pi(s) = 2$. Indeed, otherwise $\pi(s) = 3$ holds and either there are two distinct vertices $x, y \in I^\circ(u, v)$ with $d(s, x) = d(s, y) = 2$ or there exists some vertex $z \in M(u, v) - I^\circ(u, v)$ with $d(s, z) = 3$. In the first case we would get

$$F_\pi(s) - F_\pi(u) \geq \pi(x) + \pi(y) - \pi(s) \geq 2 + 2 - 3 > 0,$$

and in the second case

$$F_\pi(s) - F_\pi(u) \geq \pi(z) - \pi(s) = 4 - 3 > 0,$$

both giving a contradiction. Now, if all vertices in $M(u, v) - \{s\}$ are equidistant to s and u (as well as v), then

$$2d(s, y) \leq d(u, y) + d(v, y) \quad \text{for all } y \in M(u, v),$$

and thus $t = s$ is the required solution to the asserted inequality. Else there exists some $t \in M(u, v)$ with $d(s, t) = d(u, t) + 1$, yielding

$$0 \leq F_\pi(u) - F_\pi(s) \leq \pi(s) - \pi(t),$$

whence $\pi(t) = 2$ and $d(s, t) = 2$. Consequently,

$$d(s, y) + d(t, y) = d(u, y) + d(v, y) \quad \text{for all } y \in M(u, v),$$

and therefore s, t constitutes the required vertex pair in this case. \square

LEMMA 4.3. *Let u and v be vertices at distance 2 in a graph G for which all median sets are connected. Select a maximal subset S of $I^\circ(u, v)$ with the property that for each vertex $s \in S$ there exists a vertex $t \in S$ (not necessarily distinct from s) such that $d(s, y) + d(t, y) \leq d(u, y) + d(v, y)$ for all $y \in M(u, v)$. Then for any weight function π with support included in $J(u, v)$ there exists some vertex $w \in S$ satisfying $2F_\pi(w) \leq F_\pi(u) + F_\pi(v)$.*

Proof. By Lemma 4.2, the set S is nonempty. Now, assume the contrary and among all weight functions violating our assertion choose a function π for which the set

$$\{x \in I^\circ(u, v) - S : 2F_\pi(x) - F_\pi(u) - F_\pi(v) \leq 0\}$$

is as small as possible. This set contains some vertex x because F_π is weakly convex. Pick any $\delta > \frac{1}{2}(F_\pi(u) + F_\pi(v) - 2F_\pi(x)) \geq 0$.

First suppose that $d(x, y) = d(x, z) = 2$ for some distinct vertices $y, z \in I^\circ(u, v)$. Define a new weight function π' by

$$\pi'(w) = \begin{cases} \pi(w) + \delta & \text{if } w \in \{x, y, z\}, \\ \pi(w) & \text{otherwise.} \end{cases}$$

Then

$$F_{\pi'}(u) = F_\pi(u) + 3\delta, \quad F_{\pi'}(v) = F_\pi(v) + 3\delta, \quad F_{\pi'}(x) = F_\pi(x) + 4\delta,$$

$$F_{\pi'}(w) \geq F_\pi(w) + 3\delta \quad \text{for all } w \in I^\circ(u, v), w \neq x.$$

Hence

$$2F_{\pi'}(x) - F_{\pi'}(u) - F_{\pi'}(v) = 2F_\pi(x) - F_\pi(u) - F_\pi(v) + 2\delta > 0$$

and

$$2F_{\pi'}(w) - F_{\pi'}(u) - F_{\pi'}(v) \geq 2F_\pi(w) - F_\pi(u) - F_\pi(v)$$

for all $w \in I^\circ(u, v)$ distinct from x , contrary to the choice of π . Therefore, x is adjacent to all vertices $w \in I^\circ(u, v)$, $w \neq x$, except possibly one vertex.

Now, suppose that $d(x, z) = 3$ for some vertex $z \in M(u, v)$. Then for the modified weight function π' defined by

$$\pi'(w) = \begin{cases} \pi(z) + \delta & \text{if } w = z, \\ \pi(w) & \text{otherwise} \end{cases}$$

we have

$$F_{\pi'}(u) = F_{\pi}(u) + 2\delta, \quad F_{\pi'}(v) = F_{\pi}(v) + 2\delta, \quad F_{\pi'}(x) = F_{\pi}(x) + 3\delta,$$

$$F_{\pi'}(w) \geq F_{\pi}(w) + 2\delta \quad \text{for all } w \in I^{\circ}(u, v), w \neq x.$$

Then we obtain the same inequalities as in the preceding case and thus again arrive at a contradiction. We conclude that $d(x, z) = d(x, u) = d(x, v) = 2$ for all $z \in M(u, v) - I^{\circ}(u, v)$.

If x is adjacent to all other vertices of $I^{\circ}(u, v)$, then $2d(x, z) = d(x, u) + d(x, v)$ for all $z \in M(u, v)$, $z \neq x$, and hence we could adjoin x to S , contrary to the maximality of S . Therefore, by what has been shown, $I^{\circ}(u, v)$ contains exactly one vertex y distinct from x which is not adjacent to x . Since $x \notin S$, there exists a vertex $z \in M(u, v)$ such that $d(x, z) + d(y, z) > d(u, z) + d(v, z)$. Then necessarily $d(x, z) = d(u, z) = d(y, z) - 1 = k$ for $k \in \{1, 2\}$. Define a new weight function π' by

$$\pi'(w) = \begin{cases} \pi(w) + \delta & \text{if } w \in \{y, z\}, \\ \pi(w) & \text{otherwise.} \end{cases}$$

Then

$$F_{\pi'}(u) = F_{\pi}(u) + (k + 1)\delta, \quad F_{\pi'}(v) = F_{\pi}(v) + (k + 1)\delta,$$

$$F_{\pi'}(x) = F_{\pi}(x) + (k + 2)\delta, \quad F_{\pi'}(y) = F_{\pi}(y) + (k + 1)\delta,$$

$$F_{\pi'}(w) \geq F_{\pi}(w) + (k + 1)\delta \quad \text{for all } w \in I^{\circ}(u, v), w \neq x, y,$$

giving the same contradiction as before. This concludes the proof. \square

Now we are in a position to formulate the principal result.

THEOREM 4.4. *For a graph G all median sets are connected if and only if the following conditions are satisfied:*

- (i) *all metric triangles of G are equilateral;*
- (ii) *for any vertices u and v at distance 2 there exist a nonempty subset S of $I^{\circ}(u, v)$ and a weight function η with support included in S having the following two properties:*

(α) *every vertex $s \in S$ has a companion $t \in S$ (not necessarily distinct from s) such that*

$$d(s, x) + d(t, x) \leq d(u, x) + d(v, x) \quad \text{for all } x \in M(u, v),$$

and $\eta(s) = \eta(t)$ whenever $d(s, t) = 2$;

(β) *the joint weight of the neighbors of any $x \in J(u, v) - M(u, v)$ from S is always at least half the total weight of S :*

$$\sum_{s \in S \cap N(x)} \eta(s) \geq \frac{1}{2} \sum_{s \in S} \eta(s) \quad \text{for all } x \in J(u, v) - M(u, v).$$

Proof. First assume that G is a graph with connected medians. By Proposition 4.1 all metric triangles of G are equilateral. Let u, v be a pair of vertices at distance 2. The nonempty subset S of $I^{\circ}(u, v)$ provided by Lemma 4.3 satisfies the inequality

in (α) for each $s \in S$ and companion t . Moreover, Lemma 4.3 guarantees that for every weight function π with support included in $J(u, v)$ there exists a vertex $w \in S$ with $F_\pi(u) + F_\pi(v) - 2F_\pi(w) \geq 0$. Now, LP duality as formulated in Remark 3 for $W = S$ and $X = J(u, v)$ yields a weight function η with support included in S such that for every $x \in J(u, v)$ the weighted sum of all $d(u, x) + d(v, x) - 2d(w, x)$ ($w \in S$) is nonnegative. If $x \in J(u, v) - M(u, v)$, then

$$d(u, x) + d(v, x) - 2d(w, x) = \begin{cases} 1 & \text{if } w \in S \cap N(x), \\ -1 & \text{if } w \in S - N(x), \end{cases}$$

and therefore (β) holds. If $x \in M(u, v)$, then $d(s, x) \leq 2$ for all $s \in S$ and

$$d(u, x) + d(v, x) - 2d(w, x) = \begin{cases} 2 & \text{if } w = x \in S, \\ 0 & \text{if } w \in S \text{ with } d(w, x) = d(u, x), \\ -2 & \text{if } w \in S \text{ with } d(w, x) = 2 \text{ and } d(u, x) = 1. \end{cases}$$

In the latter case $t = x \in S$ is the companion of $s = w$. This yields the inequality $\eta(t) - \eta(s) \geq 0$, and by interchanging the role of s and t we infer that $\eta(s) = \eta(t)$ whenever $d(s, t) = 2$.

Conversely, if conditions (i) and (ii) are satisfied, then by virtue of LP duality (Remark 3) conditions (1) and (2) of Proposition 4.1 are fulfilled, whence G has connected medians. \square

COROLLARY 4.5. *All median sets in a meshed graph G are connected whenever the following condition is satisfied:*

() for any two vertices u and v with $\#I^\circ(u, v) \geq 2 = d(u, v)$ there exist (not necessarily distinct) vertices $s, t \in I^\circ(u, v)$ such that*

$$d(s, x) + d(t, x) \leq d(u, x) + d(v, x) \quad \text{for all } x \in J(u, v).$$

If, moreover, G satisfies the stronger condition requiring in addition that $d(s, t) \leq 1$, then $Med(\pi)$ induces a complete subgraph for every positive weight function π .

Proof. First observe that the inequality of $(*)$ also holds in the case that u and v are at distance 2 with a unique common neighbor s . Indeed, if $x \in J(u, v)$, then necessarily $2d(s, x) \leq d(u, x) + d(v, x)$ since G is meshed. To see that $(*)$ implies condition (ii) of Theorem 4.4, put $S = \{s, t\}$ and $\eta \equiv 1$. Then (α) is trivially satisfied. For $x \in J(u, v) - M(u, v)$ we have $d(u, x) + d(v, x) = 3$ because G is meshed, and therefore x is adjacent to at least one of s, t , thus establishing (β) .

To prove the second statement, observe that the inequality in $(*)$ actually holds for all vertices x of G . Indeed, for each vertex x select a vertex x' from $I(u, x) \cap I(v, x)$ at maximal distance to x . Then x' belongs to $J(u, v)$ and

$$\begin{aligned} d(s, x) + d(t, x) &\leq d(s, x') + d(t, x') + 2d(x, x') \\ &\leq d(u, x') + d(v, x') + 2d(x, x') \\ &= d(u, x) + d(v, x). \end{aligned}$$

Adding up all these inequalities each multiplied by the corresponding weight $\pi(x)$ then yields

$$F_\pi(s) + F_\pi(t) \leq F_\pi(u) + F_\pi(v),$$

where strict inequality holds exactly when one of the former inequalities is strict for x with $\pi(x) > 0$, for instance, when $\pi(s) > 0$. Hence, if π is positive, u and v cannot both belong to $\text{Med}(\pi)$ under these circumstances. \square

In general, we can neither dispense with weight functions nor impose any fixed upper bound on the cardinalities of the sets S occurring in Theorem 4.4, as the following example shows.

Example 1. Let G be the chordal graph with $2\binom{2k}{2} + 2k + 6$ ($k \geq 2$) vertices comprising a set $S = \{x_1, x_2, y_1, \dots, y_{2k}\}$ of mutually adjacent vertices, two vertices u and v adjacent to all of S , and vertices w_Z associated with certain subsets Z of S , namely the sets $\{x_1, x_2\}$, $\{y_1, \dots, y_{2k}\}$, and $\{x_i\} \cup Y$ for all $i = 1, 2$ and k -subsets Y of $\{y_1, \dots, y_{2k}\}$, such that each vertex w_Z is adjacent to u and all vertices from Z . Then a weight function η with support included in $I^\circ(u, v) = S$ satisfies (β) for the pair u, v if and only if for each of the above sets Z (which come in complementary pairs) one has

$$\sum_{z \in Z} \eta(z) = \sum_{y \in S - Z} \eta(y),$$

which is equivalent to

$$\eta(x_i) = k \cdot \eta(y_j) \quad \text{for all } i = 1, 2 \quad \text{and } j = 1, \dots, 2k.$$

Thus, in order to satisfy (β) , we are forced to take a weight function η having a large support on which it is not constant. Note that (α) for u, v is trivially fulfilled with any choice of $s = t$ from S . Finally, every other pair of vertices at distance 2 meets condition $(*)$ of Corollary 4.5 when either replacing u by w_Z and selecting $s = t$ from Z or replacing u, v by $w_Z, w_{Z'}$ ($Z \neq Z'$) and setting $s = t = u$. This shows that G has connected medians.

5. Particular cases. A number of graph classes [13] consist of particular meshed graphs, for instance, the classes of chordal graphs (and, more generally, bridged graphs), Helly graphs, and weakly median graphs (see below), respectively. In the case of chordal graphs the first statement of Corollary 4.5 is due to Wittenberg [30, Theorem 1.1]. Whenever a chordal graph satisfies condition $(*)$, then any pair s, t selected in $(*)$ necessarily satisfies $d(s, t) \leq 1$. Corollary 4.5 comes into full action for the class of Helly graphs. A *Helly graph* G (alias “absolute retract of reflexive graphs”) has the characteristic “Helly property” that for any vertices v_1, \dots, v_n and nonnegative integers r_1, \dots, r_n the system of inequalities $d(v_i, x) \leq r_i$ ($i = 1, \dots, n$) admits a solution x whenever $d(v_i, v_j) \leq r_i + r_j$ ($i, j = 1, \dots, n$) holds [8]. Since this Helly property for $n = 3$ implies weak convexity (as formulated in Lemma 2.1(iii)) of the function $d(\cdot, z)$, all Helly graphs are meshed. To verify condition $(*)$ (with $d(s, t) \leq 1$), first observe that the Helly property for $n = 3$ entails $J(u, v) \subseteq N(u) \cup N(v)$ and thus $M(u, v) = I^\circ(u, v)$ for every pair u, v with $d(u, v) = 2$. Now, all vertices in $(N(u) \cap J(u, v)) \cup \{v\}$ are pairwise at distance ≤ 2 , whence the Helly property guarantees a common neighbor s . Similarly, the vertices from $(N(v) \cap J(u, v)) \cup \{u\}$ have a common neighbor t . Necessarily $s, t \in I^\circ(u, v)$ and $d(s, t) \leq 1$ hold, and we have therefore established the following observation, which implies the result of Lee and Chang [22] on strongly chordal graphs (being special Helly graphs).

COROLLARY 5.1. *All median sets $\text{Med}(\pi)$ of a Helly graph G are connected, and, moreover, if π is positive, then $\text{Med}(\pi)$ induces a complete graph in G .*

In some particular classes of meshed graphs the pair s, t meeting $(*)$ can always be selected by the following trivial rule: given vertices u, v at distance 2, choose any

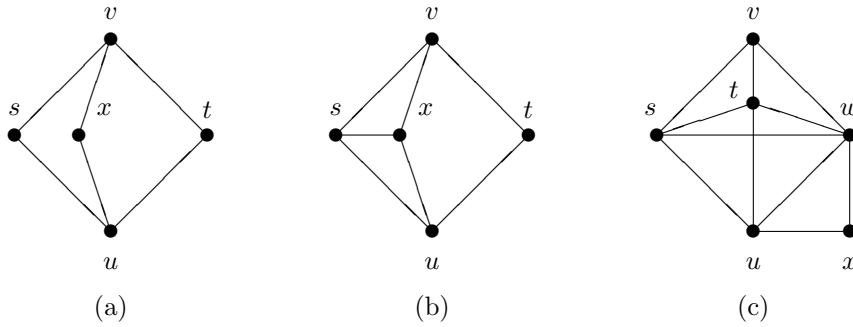


FIG. 5.1.

pair s, t from $I^\circ(u, v)$ for which $d(s, t)$ is as large as possible. Evidently, a meshed graph G satisfies condition $(*)$ with this selection rule provided that the following two requirements are met:

- (§) if $I^\circ(u, v)$ is a complete subgraph for vertices u and v at distance 2, then $d(s, x) + d(t, x) \leq d(u, x) + d(v, x)$ for all $s, t \in I^\circ(u, v)$ with $s \neq t$ and all $x \in J(u, v)$;
- (§§) if s, t, u, v induce a 4-cycle where $d(s, t) = d(u, v) = 2$, then $d(s, x) + d(t, x) = d(u, x) + d(v, x)$ for all $x \in J(u, v)$.

Notice that in (§§) we could replace the last equality by an inequality because the inequality would imply $J(s, t) = J(u, v)$ in a meshed graph and thus the reverse inequality would follow by symmetry. We will now show that the so-called weakly median graphs satisfy (§) and (§§). A graph is *weakly median* [15] if (i) any three distinct common neighbors of any two distinct vertices u and v always induce a connected subgraph and (ii) all functions $d(\cdot, z)$ satisfy the following condition, stronger than weak convexity: if u, v are at distance 2 and $|d(u, z) - d(v, z)| \leq 1$, then $2d(w, z) < d(u, z) + d(v, z)$ for some $w \in I^\circ(u, v)$. Condition (ii) is also referred to as *weak modularity*; see [4]. Trivially, weakly modular graphs are meshed.

Remark 4. A weakly modular graph G satisfies conditions (§) and (§§) if and only if G does not contain any of the graphs of Figure 5.1 as an induced subgraph. In particular, weakly median graphs have connected medians.

Proof. If G includes an induced subgraph from Figure 5.1, then s, t, u, v, x violate either (§§) or (§). Conversely, assume that G is weakly modular but violates (§) or (§§) for some vertices s, t, u, v, x . Then, as $x \in J(u, v)$, we have $d(u, x) + d(v, x) \leq 3$ by weak modularity. If $x \in I^\circ(u, v)$, then necessarily s, t, u, v, x induce one of the first two graphs in Figure 5.1. Otherwise, say, $d(u, x) + 1 = 2 = d(v, x) = d(s, x) = d(t, x)$. As G is meshed, u, v, x have some common neighbor w . If w is not adjacent to both s and t , then we are back in the preceding case with w playing the role of x . Therefore we may assume that w is a common neighbor of s and t . If s and t are not adjacent, then s, t, u, v, w induce a subgraph isomorphic to Figure 5.1(b). Otherwise, s and t are adjacent, and we obtain the graph of Figure 5.1(c) as an induced subgraph. Finally, note that the latter graph minus v as well as Figures 5.1(a)–(b) are all forbidden in a weakly median graph. \square

Inasmuch as pseudomedian graphs and quasi-median graphs are weakly median, Remark 4 (in conjunction with Proposition 3.1) generalizes some results from [3, 5, 7, 18, 27]. Another class of meshed graphs fulfilling (§) and (§§) is associated with

matroids. A *matroid* \mathcal{M} can be defined as a finite set E together with a collection \mathcal{B} of subsets (referred to as the *bases* of \mathcal{M}) such that for $B, B' \in \mathcal{B}$ and $e \in B - B'$ there exists some $e' \in B' - B$ with $(B' - \{e'\}) \cup \{e\} \in \mathcal{B}$. The *basis graph* of \mathcal{M} is the (necessarily connected) graph whose vertices are the bases of \mathcal{M} and edges are the pairs B, B' of bases differing by a single exchange. We immediately infer from the characterization established by Maurer [23] that the basis graph of every matroid is meshed (but not weakly modular in general) and satisfies (§) and (§§).

COROLLARY 5.2. *The basis graph of every matroid has connected medians.*

In view of Corollary 5.2 and Proposition 3.1 one can solve the median problem in the basis graph of a matroid $\mathcal{M} = (E, \mathcal{B})$ with the greedy algorithm. Given a weight function π on \mathcal{B} assign a weight to each element $e \in E$ by $w(e) = \sum\{\pi(B) : e \in B\}$. Applying the greedy algorithm one finds a base B^* maximizing the function $B \mapsto w(B) = \sum\{w(e) : e \in B\}$. We assert that B^* is a minimum for the function F_π in the basis graph. Indeed, if $B' = (B^* - \{e\}) \cup \{e'\}$ is any neighbor of B^* , then

$$\begin{aligned} F_\pi(B^*) - F_\pi(B') &= \sum\{\pi(B) : e \notin B, e' \in B\} - \sum\{\pi(B) : e \in B, e' \notin B\} \\ &= w(e') - w(e) \\ &= w(B') - w(B) \leq 0. \end{aligned}$$

Hence $B^* \in \text{Med}_{\text{loc}}(\pi) = \text{Med}(\pi)$, as required.

REFERENCES

[1] T.T. ARKHIPOVA AND I.V. SERGIENKO, *On conditions of coincidence of local and global extrema in optimization problems*, Kibernetika, 1 (1976), pp. 13–15 (in Russian).
 [2] H.-J. BANDELDT, *NP-completeness for non-global local subgraph medians*, Discrete Appl. Math., submitted.
 [3] H.-J. BANDELDT AND J.P. BARTHÉLEMY, *Medians in median graphs*, Discrete Appl. Math., 8 (1984), pp. 131–142.
 [4] H.-J. BANDELDT AND V. CHEPOI, *A Helly theorem in weakly modular space*, Discrete Math., 126 (1996), pp. 25–39.
 [5] H.-J. BANDELDT AND H.M. MULDER, *Pseudo-median graphs: Decomposition via amalgamation and Cartesian multiplication*, Discrete Math., 94 (1991), pp. 161–180.
 [6] H.-J. BANDELDT, H.M. MULDER, AND V. SOLTAN, *Weak Cartesian factorization with icosahedra, 5-wheels, and subhyperoctahedra as factors*, J. Graph Theory, submitted.
 [7] H.-J. BANDELDT, H.M. MULDER, AND E. WILKEIT, *Quasi-median graphs and algebras*, J. Graph Theory, 18 (1994), pp. 681–703.
 [8] H.-J. BANDELDT AND E. PESCH, *Dismantling absolute retracts of reflexive graphs*, European J. Combin., 10 (1989), pp. 211–220.
 [9] H.-J. BANDELDT AND M. VAN DE VEL, *A fixed cube theorem for median graphs*, Discrete Math., 67 (1987), pp. 129–137.
 [10] J.P. BARTHÉLEMY AND M.F. JANOWITZ, *A formal theory of consensus*, SIAM J. Discrete Math., 4 (1991), pp. 305–322.
 [11] J.P. BARTHÉLEMY, B. LECLERC, AND B. MONJARDET, *On the use of ordered sets in problems of comparison and consensus of classifications*, J. Classification, 3 (1986), pp. 187–224.
 [12] J.P. BARTHÉLEMY AND B. MONJARDET, *The median procedure in cluster analysis and social choice theory*, Math. Social Sci., 1 (1981), pp. 235–268.
 [13] A. BRANDSTÄDT, V.B. LE, AND J.P. SPINRAD, *Graph Classes: A Survey*, SIAM, Philadelphia, 1999.
 [14] H. BUSEMANN, *The Geometry of Geodesics*, Academic Press, New York, 1955.
 [15] V. CHEPOI, *Separation of two convex sets in convexity structures*, J. Geom., 50 (1994), pp. 30–51.
 [16] V. CHEPOI, *On distance-preserving and domination elimination orderings*, SIAM J. Discrete Math., 11 (1998), pp. 414–436.
 [17] V. CHVÁTAL, *Linear Programming*, Freeman, New York, 1983.
 [18] F. DRAGAN AND V. CHEPOI, *Medians of pseudo-median graphs*, Oper. Res. Manage. Sci. (Kiev), 35 (1991), pp. 47–56 (in Russian).

- [19] T. FEDER, *Stable networks and product graphs*, Mem. Amer. Math. Soc., 116 (1995).
- [20] G.R.T. HENDRY, *On graphs with prescribed median I*, J. Graph Theory, 9 (1985), pp. 477–481.
- [21] T.T. LEBEDEVA, I.V. SERGIENKO, AND V. SOLTAN, *On conditions for the coincidence of local and global extrema in problems of discrete optimization*, Kibernetika, 5 (1984), pp. 58–65 (in Russian).
- [22] H.-Y. LEE AND G.J. CHANG, *The w -median of a connected strongly chordal graph*, J. Graph Theory, 18 (1994), pp. 673–680.
- [23] S.B. MAURER, *Matroid basis graphs I*, J. Combinatorial Theory Ser. B, 14 (1973), pp. 216–240.
- [24] F.R. MCMORRIS AND R.C. POWERS, *The median procedure in a formal theory of consensus*, SIAM J. Discrete Math., 8 (1995), pp. 507–516.
- [25] J. NIEMINEN, *Distance center and centroid of a median graph*, J. Franklin Inst., 323 (1987), pp. 89–94.
- [26] P.J. SLATER, *Medians of arbitrary graphs*, J. Graph Theory, 4 (1980), pp. 389–392.
- [27] P. SOLTAN AND V. CHEPOI, *The solution of the Weber problem for discrete median metric spaces*, Trudy Tbilisskogo Mat. Inst., 85 (1987), pp. 52–76 (in Russian).
- [28] B.C. TANSEL, R.L. FRANCIS, AND T.J. LOWE, *Location on networks: A survey. I*, Management Sci., 29 (1983), pp. 482–497.
- [29] B.C. TANSEL, R.L. FRANCIS, AND T.J. LOWE, *Location on networks: A survey. II*, Management Sci., 29 (1983), pp. 498–511.
- [30] H. WITTENBERG, *Local medians in chordal graphs*, Discrete Appl. Math., 28 (1990), pp. 287–296.

FINDING 1-FACTORS IN BIPARTITE REGULAR GRAPHS AND EDGE-COLORING BIPARTITE GRAPHS*

ROMEO RIZZI†

Abstract. This paper gives a new and faster algorithm to find a 1-factor in a bipartite Δ -regular graph. The time complexity of this algorithm is $\mathcal{O}(n\Delta + n \log n \log \Delta)$, where n is the number of nodes. This implies an $\mathcal{O}(n \log n \log \Delta + m \log \Delta)$ algorithm to edge-color a bipartite graph with n nodes, m edges, and maximum degree Δ .

Key words. time-tabling, edge-coloring, perfect matching, regular bipartite graphs

AMS subject classifications. Primary, 05C70 ; Secondary, 05C85

PII. S0895480199351136

1. Introduction. Let G be a bipartite regular graph. A celebrated result of König [5] (see [6] for a compact proof) states that G can be *factorized*; that is, $E(G)$ can be decomposed as the union of edge-disjoint 1-factors. (A 1-factor is simply another way to say perfect matching.) Any bipartite matching algorithm can thus be employed to find a 1-factor in G and hence to factorize G . However, there exist faster methods exploiting the regularity of G . Cole and Hopcroft [1] gave an $\mathcal{O}(n\Delta + n \log n \log^2 \Delta)$ algorithm to find a 1-factor in a Δ -regular bipartite graph with n nodes. Schrijver [7] gave an $\mathcal{O}(n\Delta^2)$ algorithm for the same problem. Depending on the relative values of Δ and n , either algorithm gives the best-so-far proven worst-case asymptotic bound. We do not know of any randomized algorithm with better bounds.

In section 2, we give an $\mathcal{O}(n\Delta + n \log n \log \Delta)$ deterministic algorithm, thus improving the bound on the side of Cole and Hopcroft's.

Let G be a bipartite graph (possibly not regular) with n nodes, m edges, and maximum degree Δ . An edge-coloring of G assigns to each edge of G one of Δ possible colors so that no two adjacent edges receive the same color. By a simple reduction, the above cited result of König [5] implies that every bipartite graph admits an edge-coloring. Kapoor and Rizzi [4] gave an algorithm to edge-color G in $T_{n,m,\Delta} + \mathcal{O}(m \log \Delta)$ time, where $T_{n,m,\Delta}$ is the time needed to find a 1-factor in a d -regular bipartite graph with $\mathcal{O}(m)$ edges, $\mathcal{O}(n)$ nodes, and $d \leq \Delta$. Motivated by this result, we investigated Cole and Hopcroft's 1-factor algorithm for possible improvements. This effort culminated in the new and faster 1-factor procedure given in this paper. Combining this 1-factor procedure with the edge-coloring algorithm given in [4] we can edge-color G in $\mathcal{O}(n \log n \log \Delta + m \log \Delta)$ time.

2. The algorithm. Our graphs have no loops but possibly have parallel edges. A graph without parallel edges is said to be *simple*. The *support* of a graph \mathcal{G} is a simple graph G with $V(G) = V(\mathcal{G})$ and such that two nodes are adjacent in G if and only if they are adjacent in \mathcal{G} . The input of our algorithm is a bipartite Δ -regular

*Received by the editors January 27, 1999; accepted for publication (in revised form) March 22, 2002; published electronically May 8, 2002. This research was carried out with the financial support of project TMR-DONET grant ERB FMRX-CT98-0202 of the European Community and was partially supported by a postdoctorate fellowship by the Dipartimento di Matematica Pura ed Applicata of the University of Padova, 35131 Padova, Italy.

<http://www.siam.org/journals/sidma/15-3/35113.html>

†Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, 38050 Povo, Italy (romeo@science.unitn.it).

graph \mathcal{G}_0 with n nodes and $m = \frac{n}{2}\Delta$ edges. We encode a graph \mathcal{G} by giving its support G and by specifying for every edge uv of G the number $g[uv]$ of edges in \mathcal{G} having u and v as endnodes. The number $g[uv]$ is a *positive* integer, called the *multiplicity* of edge uv in \mathcal{G} . Throughout the following, we should keep in mind that the proposed algorithms deal with graphs by actually manipulating supports and multiplicities.

In general, whenever \mathcal{X} denotes a graph, then X stands for the support of \mathcal{X} and x for the multiplicities' vector of \mathcal{X} . Even if no value $x[uv] = 0$ is stored explicitly by the algorithm, we will consider $x[uv]$ to be 0 when u and v are not adjacent in \mathcal{X} . All graphs considered are restricted to have the same node set V , namely $V = V(\mathcal{G}_0)$. The *sum* $\mathcal{G} + \mathcal{H}$ of two graphs \mathcal{G} and \mathcal{H} is the graph \mathcal{S} with $s = g + h$ (componentwise). The maximum degree of a node in a graph \mathcal{H} is denoted by $\Delta(\mathcal{H})$. Throughout the whole algorithm the value Δ will also be a constant and stands for $\Delta(\mathcal{G}_0)$.

We say that graph \mathcal{G} *contains* graph \mathcal{H} when $E(\mathcal{H}) \subseteq E(\mathcal{G})$. When \mathcal{G} contains \mathcal{H} (in short, $\mathcal{H} \subseteq \mathcal{G}$) and \mathcal{H} contains a 1-factor, then \mathcal{G} also contains a 1-factor. Our algorithm will modify the input graph \mathcal{G}_0 , thus determining a sequence $\mathcal{G}_0, \mathcal{G}_1, \dots$ of graphs. Each graph in the sequence will be contained in the previous one, and all graphs will be regular. The support of the last graph in the sequence will be a 1-factor.

A graph \mathcal{G} is said to be *sparse* if $|E(\mathcal{G})| \leq 2n \log \Delta$. For our manipulations to be performed efficiently it will be crucial to assume we are working on sparse graphs. Thus a first phase of our algorithm will have to make \mathcal{G}_0 sparse. Subsection 2.1 describes a preprocessing algorithm to sparsify \mathcal{G}_0 . This preprocessing algorithm was first proposed by Cole and Hopcroft in [1]. Here we prefer to describe it in some more detail.

2.1. Why we assume \mathcal{G}_0 to be sparse: The preprocessing phase. Cole and Hopcroft [1] proposed the following method to obtain a sparse Δ -regular graph \mathcal{H} contained in a Δ -regular graph \mathcal{G} . The method takes $\mathcal{O}(m)$ time.

Obviously, $g[e] \leq \Delta$ for every $e \in E(\mathcal{G})$. Let $k = \lceil \log \Delta \rceil + 1$ and let $g[e]_{[k]}, \dots, g[e]_{[1]}, g[e]_{[0]}$ be the binary encoding of $g[e]$. This means that $g[e] = \sum_{i=0}^k g[e]_{[i]} \cdot 2^i$. For $i = 0, 1, \dots, k$ define the edge-set

$$E_i(\mathcal{G}) = \{e \in E(\mathcal{G}) : g[e]_{[i]} = 1\}.$$

For example, $E_0(\mathcal{G})$ is the set of edges having odd multiplicity in \mathcal{G} .

Start with $\mathcal{H} = \mathcal{G}$. When each $E_i(\mathcal{H})$ is acyclic, then $|E_i(\mathcal{H})| < n$ for $i = 1, \dots, k$; hence \mathcal{H} is sparse. The idea is to first make $E_0(\mathcal{H})$ acyclic, then $E_1(\mathcal{H})$, and so on, until $E_k(\mathcal{H})$. Let C be a cycle contained in $E_{\bar{i}}(\mathcal{H})$ with \bar{i} as small as possible. Let M_1, M_2 be two matchings such that $C = M_1 \cup M_2$. Then, by setting $h[e] \leftarrow h[e] - 2^{\bar{i}}$ for every edge e in M_1 and $h[e] \leftarrow h[e] + 2^{\bar{i}}$ for every edge e in M_2 , we do not affect any of $E_0(\mathcal{H}), E_1(\mathcal{H}), \dots, E_{\bar{i}-1}(\mathcal{H})$ but reduce $|E_{\bar{i}}(\mathcal{H})|$ by $|C|$. Note that this manipulation preserves the Δ -regularity of \mathcal{H} . Moreover, the graph produced by the manipulation will be contained in the one it has been obtained from. This preprocessing algorithm can be implemented to run in time $\mathcal{O}(m + \frac{m}{2} + \frac{m}{4} + \dots) = \mathcal{O}(m)$. We close this subsection with two more implementational subtleties.

1. After setting $h[e] \leftarrow h[e] - 2^{\bar{i}}$ we check if $h[e] < 2^{\bar{i}}$. If this is the case, then $e \notin E_j$ for any $j > \bar{i}$ and edge e is removed from the “working input graph” and is placed in the “definitive graph.” The “definitive graph” is output when the procedure terminates.

2. The search for circuit C is done as follows. Starting from a node v_o construct a depth-first search tree T , and, when a circuit C is detected, then all nodes of the

tree but not in C which have a node of C as ancestor are guaranteed not to belong to any circuit in $E_{\bar{v}}(\mathcal{H})$, so we discard them and free the nodes in $V(C)$ after performing the above described manipulation. All the other nodes remain in the tree. When T is completed, then we can discard all nodes in $V(T)$ and construct a new depth-first search tree starting from any (not-yet-discarded) node. When no node is left, then $E_{\bar{v}}$ is acyclic.

2.2. Why we assume Δ to be odd: Procedure *EulerSplit*. The reduction given in this subsection dates back to Gabow [2].

A graph \mathcal{G} is called *Eulerian* when every node has even degree in \mathcal{G} . We first describe a basic procedure, called *EulerSplit*, which takes as input an Eulerian graph \mathcal{G} and returns as output a graph \mathcal{H} with $h \leq g$ (componentwise) such that for every node $v \in V$ the degree of v in \mathcal{G} is twice the degree of v in \mathcal{H} . From the following description, Procedure *EulerSplit* can be implemented as to take $\mathcal{O}(n \log \Delta)$ time, when \mathcal{G} is sparse.

Decompose G as $G_e + G_o$, where G_o contains precisely those edges of G which have odd multiplicity in \mathcal{G} . Since \mathcal{G} is Eulerian, then G_o is Eulerian. By orienting the edges of G_o in the direction they are traversed by an Euler tour we find an orientation of G_o such that the in-degree equals the out-degree for every node. Now we decompose G_o as $\overleftarrow{G}_o + \overrightarrow{G}_o$, where \overrightarrow{G}_o contains precisely those edges of G_o which have been oriented as to go from, say, the “left” side of the bipartition to the “right” side. Consider the graph \mathcal{H} contained in \mathcal{G} and such that

$$h[e] = \frac{g[e]}{2} \quad \text{if } e \text{ is an edge of } G_e, \quad \begin{cases} h[e] = \lfloor \frac{g[e]}{2} \rfloor & \text{if } e \text{ is an edge of } \overleftarrow{G}_o, \\ h[e] = \lceil \frac{g[e]}{2} \rceil & \text{if } e \text{ is an edge of } \overrightarrow{G}_o. \end{cases}$$

Note that $h \leq g$ and for every node $v \in V$ the degree of v in \mathcal{G} is twice the degree of v in \mathcal{H} .

The reason why we can always assume Δ to be odd is the following procedure.

Procedure 1 MAKEODD (\mathcal{G})	(precondition: \mathcal{G} is regular)
<ol style="list-style-type: none"> 1. if $\Delta(\mathcal{G})$ is odd, then return \mathcal{G}; 2. else return <i>MakeOdd(EulerSplit</i>(\mathcal{G})). 	

2.3. Procedure *Split* and taking complements. Our algorithm calls Procedure *Split*, an important operation due to Cole and Hopcroft [1].

A graph \mathcal{S} is a *slice* of a graph \mathcal{G} when $s \leq g$. Slice \mathcal{S} is *big* when $|E(\mathcal{G})| \leq 2|E(\mathcal{S})|$. For $k \geq 1$, slice \mathcal{S} is a $(k, k+1)$ -*slice* if each node $v \in V$ has degree either k or $k+1$ in \mathcal{S} . We denote by $odd(\mathcal{S})$ the set of those nodes having odd degree in \mathcal{S} . The *complement* of a $(k, k+1)$ -slice \mathcal{S} in \mathcal{G} is the unique graph \mathcal{T} such that $\mathcal{S} + \mathcal{T} = \mathcal{G}$. Note that \mathcal{T} is a $(\Delta - k - 1, \Delta - k)$ -slice. Moreover, when Δ is odd, then $odd(\mathcal{T}) = V \setminus odd(\mathcal{S})$. When \mathcal{G} is sparse, the complement can be computed in $\mathcal{O}(n \log \Delta)$ time.

Procedure *Split* takes as input a $(k, k+1)$ -slice \mathcal{S} of \mathcal{G} and returns an $(h, h+1)$ -slice \mathcal{S}' of \mathcal{G} with $|odd(\mathcal{S}')| \leq \frac{|odd(\mathcal{S})|}{2}$. The computation of $\mathcal{S}' = Split(\mathcal{S}; \mathcal{G})$ is accomplished as follows. Decompose \mathcal{S} as $\mathcal{S}_e + \mathcal{S}_o$, where \mathcal{S}_o contains precisely those edges of \mathcal{S} which have odd multiplicity in \mathcal{S} . Orient the edges of \mathcal{S}_o so that for every node the in-degree differs from the out-degree by at most 1. When \mathcal{G} is sparse, this can be done in $\mathcal{O}(n \log \Delta)$ time by, for example, adding some artificial edges to \mathcal{S}_o as to make it

Eulerian and then proceeding as in subsection 2.2. Decompose S_o as $\overleftarrow{S}_o + \overrightarrow{S}_o$ as explained in subsection 2.2. Let w be the odd value in $\{k, k + 1\}$. If $w = k$, then let S_o^{up} be a big slice of S_o in $\{\overleftarrow{S}_o, \overrightarrow{S}_o\}$ and let S_o^{down} be the other slice. Otherwise let S_o^{down} be a big slice of S_o in $\{\overleftarrow{S}_o, \overrightarrow{S}_o\}$ and let S_o^{up} be the other slice. Consider the graph \mathcal{P} contained in \mathcal{S} and such that

$$p[e] = \frac{s[e]}{2} \quad \text{if } e \text{ is an edge of } S_e, \quad \begin{cases} p[e] = \left\lceil \frac{s[e]}{2} \right\rceil & \text{if } e \text{ is an edge of } S_o^{up}, \\ p[e] = \left\lfloor \frac{s[e]}{2} \right\rfloor & \text{if } e \text{ is an edge of } S_o^{down}. \end{cases}$$

If $w = k + 1$, then \mathcal{P} is a $(\frac{k}{2}, \frac{k}{2} + 1)$ -slice where at most $\frac{|odd(\mathcal{S})|}{2}$ nodes have degree $\frac{k}{2} + 1$. Therefore, if $\frac{k}{2} + 1$ is odd, then $\mathcal{S}' = \mathcal{P}$ will work, and otherwise we will take as \mathcal{S}' the complement of \mathcal{P} . If $w = k$, then \mathcal{P} is a $(\frac{k+1}{2} - 1, \frac{k+1}{2})$ -slice where at most $\frac{|odd(\mathcal{S})|}{2}$ nodes have degree $\frac{k+1}{2}$. Therefore, if $\frac{k+1}{2}$ is odd, then $\mathcal{S}' = \mathcal{P}$ will work, and otherwise we will take as \mathcal{S}' the complement of \mathcal{P} . Note that when \mathcal{G} is sparse, then *Split* requires $\mathcal{O}(n \log \Delta)$ time.

2.4. The algorithm of Cole and Hopcroft. The following pseudocode describes a simplified version¹ of Cole and Hopcroft’s algorithm [1].

Algorithm 2 COLE_HOPCROFT (\mathcal{G}_0) (precondition: \mathcal{G}_0 is Δ -regular)

1. $\mathcal{G} \leftarrow \text{MakeOdd}(\mathcal{G}_0)$;
2. while G is not a 1-factor *invariant*: $\mathcal{G} \subseteq \mathcal{G}_0$ is regular with $\Delta(\mathcal{G})$ odd
3. $\mathcal{S} \leftarrow \mathcal{G}$;
4. do $\mathcal{S} \leftarrow \text{Split}(\mathcal{S}; \mathcal{G})$;
5. while $odd(\mathcal{S})$ is not empty; *invariant*²: \mathcal{S} is a $(k, k + 1)$ -slice of \mathcal{G}
6. $\mathcal{G} \leftarrow \text{MakeOdd}(\mathcal{S})$;
7. return G .

Loop 4–5, when entered, cycles $\mathcal{O}(\log n)$ times, since $odd(\mathcal{S})$ is at least halved each time. Loop 2–6, when entered, cycles $\mathcal{O}(\log \Delta)$ times, since $\Delta(\mathcal{G})$ is at least halved each time. All operations involved in loop 2–6, except *MakeOdd*, cost $\mathcal{O}(n \log \Delta)$, since by section 2.1 we can assume that \mathcal{G}_0 is sparse. Since *EulerSplit* is executed $\mathcal{O}(\log \Delta)$ times, the total time spent in *MakeOdd* over the whole execution of the algorithm is $\mathcal{O}(n \log^2 \Delta)$. Hence Cole and Hopcroft’s algorithm is $\mathcal{O}(n\Delta + n \log n \log^2 \Delta)$.

2.5. Our starting point: Procedure Starter. Our starting point is essentially the inner loop in Cole and Hopcroft’s algorithm. We have just shown its cost to be $\mathcal{O}(n \log n \log \Delta)$ for sparse input graphs. Here we assume Δ to be odd.

Procedure 3 STARTER (\mathcal{G}) (precondition: \mathcal{G} is Δ -regular and Δ is odd)

1. $\mathcal{S} \leftarrow \mathcal{G}$;
2. do $\mathcal{S} \leftarrow \text{Split}(\mathcal{S}; \mathcal{G})$;
3. while $odd(\mathcal{S})$ is not empty; *invariant*²: \mathcal{S} is a $(k, k + 1)$ -slice of \mathcal{G}
4. return \mathcal{S} .

The output \mathcal{S} of Procedure *Starter* is a δ -regular graph contained in \mathcal{G} . A crucial property about \mathcal{S} and \mathcal{G} is that δ and Δ are *coprime*; that is, the only integer which

¹In the original version step 6 assigns to \mathcal{G} the complement of \mathcal{S} in \mathcal{G} , in case \mathcal{S} is a big slice of \mathcal{G} .

divides both is 1. Indeed, regarding \mathcal{G} as a $(\Delta - 1, \Delta)$ -slice of \mathcal{G} , then $\mathcal{S} = \text{Split}(\mathcal{G}; \mathcal{G})$ is a $(\frac{\Delta-1}{2}, \frac{\Delta+1}{2})$ -slice of \mathcal{G} , that is, a $(k, k + 1)$ -slice where both k and $k + 1$ are coprime with Δ . A second invariant² of loop 2–3 in Procedure *Starter* is that the even value among k and $k + 1$ is coprime with Δ . In fact, $\text{g.c.d.}(a, b) = \text{g.c.d.}(a, a - b)$ (taking complement) and $\text{g.c.d.}(a, 2b) = \text{g.c.d.}(a, b)$, assuming that a is odd.

The next subsection describes an algorithm, which, given as input a Δ -regular graph \mathcal{G} and a δ -regular graph \mathcal{S} , returns a regular graph \mathcal{F} with $f \leq g + s$ and $\Delta(\mathcal{F}) = \text{g.c.d.}(\Delta; \delta)$ in $\mathcal{O}((|E(\mathcal{G})| + |E(\mathcal{S})|) \log^2 \Delta)$ time. In our case $s \leq g$ and $\text{g.c.d.}(\Delta, \delta) = 1$; hence a 1-factor of \mathcal{G} is returned. Moreover, $|E(\mathcal{S})| < |E(\mathcal{G})| = \mathcal{O}(n \log \Delta)$, and the time bound is $\mathcal{O}(n \log^3 \Delta)$. This term is dominated by the $\mathcal{O}(m)$ cost of the preprocessing phase.

2.6. Computing the g.c.d. by sums and shiftings. When a and b are two positive integers we denote by $\text{g.c.d.}(a, b)$ the greatest common divisor of a and b . When both a and b are even, then $\text{g.c.d.}(a, b) = 2 \text{g.c.d.}(\frac{a}{2}, \frac{b}{2})$. This section considers an algorithm to compute $\text{g.c.d.}(a, b)$ when at least one of a and b is odd. The procedure is allowed to use the following operations: dividing an even by 2 (this corresponds to *EulerSplit* and costs $\mathcal{O}(n \log \Delta)$), testing evenness, summing two integers (the sum of two graphs also costs $\mathcal{O}(n \log \Delta)$), and comparing two integers (greater, less, or equal?). The procedure goes as follows: When one of the two numbers is even, then we divide it by 2, and the g.c.d. does not change since the other number is odd. So both numbers are odd. Therefore their sum σ is even, and, if we substitute the biggest of the two numbers by $\frac{\sigma}{2}$, the g.c.d. does not change. Eventually the two numbers will be equal. However, now $\text{g.c.d.}(a, a) = a$.

We now show that the above procedure³ uses $\mathcal{O}(\log^2(a + b))$ operations. This is because each time $\frac{\sigma}{2}$ is even, then σ actually decreases at least by a factor of $\frac{3}{4}$, and, when $\frac{\sigma}{2}$ is odd, then $|b - a|$ decreases at least by a factor of 2, while σ is never increased.

Here is the algorithm promised in the end of the previous subsection:

Algorithm 4 G.C.D. $(\mathcal{G}, \mathcal{S})$ (precondition: \mathcal{G} and \mathcal{S} are regular)

1. $\mathcal{G} \leftarrow \text{MakeOdd}(\mathcal{G}); \mathcal{S} \leftarrow \text{MakeOdd}(\mathcal{S});$
2. while $\Delta(\mathcal{G}) \neq \Delta(\mathcal{S})$
3. by eventually exchanging \mathcal{G} and \mathcal{S} , assume $\Delta(\mathcal{G}) \geq \Delta(\mathcal{S});$
4. $\tilde{\mathcal{G}} \leftarrow \text{MakeOdd}(\mathcal{G} + \mathcal{S}).$

Acknowledgment. I thank the referee for the detailed feedback he has given.

REFERENCES

[1] R. COLE AND J. HOPCROFT, *On edge coloring bipartite graphs*, SIAM J. Comput., 11 (1982), pp. 540–546.
 [2] H.N. GABOW, *Using Euler partitions to edge color bipartite multigraphs*, Internat. J. Comput. Information Sci., 5 (1976), pp. 345–355.
 [3] H.N. GABOW AND O. KARIV, *Algorithms for edge coloring bipartite graphs and multigraphs*, SIAM J. Comput., 11 (1982), pp. 117–129.
 [4] A. KAPOOR AND R. RIZZI, *Edge-coloring bipartite graphs*, J. Algorithms, 34 (2000), pp. 390–396.

²Second invariant: Δ is coprime with the even value among k and $k + 1$.

³A deeper analysis of a related and similar procedure is given in [4]

- [5] D. KÖNIG, *Graphok és alkalmazásuk a determinánsok és a halmazok elméletére*, Matematikai és Természettudományi Értesítő, 34 (1916), pp. 104–119 (in Hungarian).
- [6] R. RIZZI, *König's edge coloring theorem without augmenting paths*, J. Graph Theory, 29 (1998), p. 87.
- [7] A. SCHRIJVER, *Bipartite edge coloring in $\mathcal{O}(\Delta m)$ time*, SIAM J. Comput., 28 (1998), pp. 841–846.

INFINITE FAMILIES OF 3-DESIGNS FROM \mathbf{Z}_4 -GOETHALS CODES WITH BLOCK SIZE 8*

KALLE RANTO[†]

Abstract. We construct several new families of simple 3-designs from codewords of the \mathbf{Z}_4 -Goethals codes. These designs have parameters $3-(2^m, 8, \lambda)$ with odd $m \geq 5$. The smallest design has $\lambda = 14(2^m - 8)/3$, and the others are corollaries of this and some previously known designs. In the existence proofs we analyze the low-weight codewords and count the number of solutions to certain systems of equations over finite fields.

Key words. t -design, Goethals code, quaternary code, Kloosterman sum

AMS subject classifications. 05B05, 94B05, 11T23

PII. S0895480101385882

1. Introduction. A t - (v, k, λ) design is a pair (X, B) , where X is a v -element set of *points* and B is a collection of k -element subsets of X (called *blocks*) with the property that every t -element subset of X is contained in exactly λ blocks. A design is *simple* if all the blocks are distinct; otherwise, the design is said to have *repeated blocks*. In this paper all designs considered are simple unless otherwise stated.

Let \mathbf{Z}_4 denote the ring of integers modulo 4. Research on constructing t -designs from error-correcting codes over \mathbf{Z}_4 started with articles [6, 8] where it was shown by computer searches that in the \mathbf{Z}_4 -Golay code the supports of codewords of Hamming weight 10 and 12 yield 5- $(24, 10, 36)$ and 5- $(24, 12, 1584)$ designs, respectively. These results were later proved analytically in three different ways [4, 20, 21].

In addition to the above interesting but restricted designs, the codewords of Hamming weight 5 and 6 in the \mathbf{Z}_4 -Preparata codes have been used to construct infinite families of 3-designs [10, 11]. Also, the supports of the \mathbf{Z}_4 -Kerdock codes contain infinite families of 3-designs [22]. For a survey on t -designs and \mathbf{Z}_4 -codes, see [12].

From the \mathbf{Z}_4 -Goethals code Shin, Kumar, and Helleseeth [18] constructed a $3-(2^m, 7, 14(2^m - 8)/3)$ design for odd $m \geq 5$ by taking the supports of codewords of Hamming weight 7. In this paper we introduce the notion of a *lifting rank* and classify the supports of codewords of Hamming weight 8 by their lifting rank and symmetrized weight. The supports in one class define a $3-(2^m, 8, 14(2^m - 8)/3)$ design. It is interesting that this design has exactly the same parameters as the design mentioned above except the block size which is equal to 8. This special feature is explained in section 8 with affine geometry.

There have also been attempts to give some general theorems to decide whether the supports of some codewords over \mathbf{Z}_4 would form a t -design. The results are usually more complicated than the celebrated Assmus–Mattson theorem [1] in the case of linear codes over finite fields. An Assmus–Mattson-type theorem by Shin, Kumar, and Helleseeth [19] does not imply our main theorem, but it is used to derive Corollary 5.4. We also claim that the theorem by Tanabe [21] does not give any designs from

*Received by the editors March 7, 2001; accepted for publication (in revised form) March 22, 2002; published electronically May 8, 2002.

<http://www.siam.org/journals/sidma/15-3/38588.html>

[†]Turku Centre for Computer Science TUCS, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland. Current address: Department of Mathematics, University of Turku, FIN-20014 Turku, Finland (kara@utu.fi, <http://www.tucs.fi/>).

the \mathbf{Z}_4 -Goethals code. The required calculations are messy and not included here; see also comments in [12].

We start by recalling some preliminary results, and in section 3 we define the \mathbf{Z}_4 -Goethals codes and analyze the low-weight codewords. In section 4 we list some relevant known designs and then in section 5 the new designs. The proofs are postponed to sections 6 and 7. In section 8 we describe the connection mentioned above, and in section 9 we come finally to conclusions.

2. Preliminaries. Let \mathbf{F} be the finite field with $n = 2^m$ elements and $\text{Tr}(x)$ the trace function from \mathbf{F} to the binary field \mathbf{F}_2 . The following lemma is well known.

LEMMA 2.1. *The quadratic equation $x^2 + x = a$ with $a \in \mathbf{F}$ has two roots in \mathbf{F} , if $\text{Tr}(a) = 0$, and no roots in \mathbf{F} if $\text{Tr}(a) = 1$.*

Equation $x^2 + bx = a$, where $b \neq 0$, can be transformed to $(x/b)^2 + x/b = a/b^2$, and the condition in the previous lemma changes to $\text{Tr}(a/b^2) = 0$.

With [3, Theorem 2] it is simple to count the following result.

LEMMA 2.2. *Let m be odd. The cubic equation $x^3 + x = a$ has three roots in \mathbf{F} for $(n - 2)/6$ values of $a \in \mathbf{F}^*$.*

The Kloosterman sums are closely related to the \mathbf{Z}_4 -Goethals codes (see [13, 14]), but we need them only in the form of the next theorem. We denote the sets $\mathbf{F} \setminus \{0\}$ and $\mathbf{F} \setminus \{0, 1\}$ by \mathbf{F}^* and \mathbf{F}^{**} , respectively.

DEFINITION 2.3. *The Kloosterman sum $K(a)$ for $a \in \mathbf{F}^*$ is defined as*

$$K(a) = \sum_{\eta \in \mathbf{F}^*} (-1)^{\text{Tr}(\eta) + \text{Tr}(\frac{a}{\eta})}.$$

THEOREM 2.4. *Let m be odd. For every $a \in \mathbf{F}^{**}$*

$$K(a^3(a + 1)) = K(a(a + 1)^3).$$

Proof. Shin, Kumar, and Helleseth [18] have proved the identity

$$K\left(\frac{a}{1 + a^4}\right) = K\left(\frac{a^3}{1 + a^4}\right)$$

for every $a \in \mathbf{F}^{**}$ and odd m . By substituting $a \mapsto a/(1 + a)$ in this identity we get the claim. The substitution is clearly a permutation $\mathbf{F}^{**} \rightarrow \mathbf{F}^{**}$. \square

We consider linear \mathbf{Z}_4 -codes of length n which are subgroups of \mathbf{Z}_4^n with componentwise addition. Let $\mathbf{c} = (c_1, \dots, c_n)$ be a codeword.

DEFINITION 2.5. *The support of \mathbf{c} is $\chi(\mathbf{c}) = \{k \mid c_k \neq 0\}$ and the multiplicity of $i \in \mathbf{Z}_4$ in \mathbf{c} is $n_i(\mathbf{c}) = |\{k \mid c_k = i\}|$. We define complete, symmetrized, Lee weight, and Hamming weight enumerators of \mathbf{c} as*

$$\begin{aligned} \text{cwe}(\mathbf{c}) &= W^{n_0(\mathbf{c})} X^{n_1(\mathbf{c})} Y^{n_2(\mathbf{c})} Z^{n_3(\mathbf{c})}, & \text{swe}(\mathbf{c}) &= W^{n_0(\mathbf{c})} X^{n_1(\mathbf{c}) + n_3(\mathbf{c})} Y^{n_2(\mathbf{c})}, \\ \text{lwe}(\mathbf{c}) &= W^{n_0(\mathbf{c})} X^{n_1(\mathbf{c}) + 2n_2(\mathbf{c}) + n_3(\mathbf{c})}, & \text{hwe}(\mathbf{c}) &= W^{n_0(\mathbf{c})} X^{n_1(\mathbf{c}) + n_2(\mathbf{c}) + n_3(\mathbf{c})}, \end{aligned}$$

and of a code C , for example, as $\text{cwe}(C) = \sum_{\mathbf{c} \in C} \text{cwe}(\mathbf{c})$.

The variable W is usually unnecessary except with the zero word W^n .

DEFINITION 2.6. *A Gray map $\phi : \mathbf{Z}_4^n \rightarrow \mathbf{F}_2^{2n}$ is defined for one coordinate as*

$$\phi(0) = 00, \quad \phi(1) = 10, \quad \phi(2) = 11, \quad \phi(3) = 01$$

with labels $\phi(c_i) = (c_{iL}, c_{iR})$ and for the whole codeword as

$$\phi(\mathbf{c}) = (c_{1L}, c_{2L}, \dots, c_{nL} \mid c_{1R}, c_{2R}, \dots, c_{nR}) = (\mathbf{c}_L \mid \mathbf{c}_R).$$

3. \mathbf{Z}_4 -Goethals codes. Let $R = \text{GR}(4, m)$ be a Galois ring of characteristic 4 with $n^2 = 4^m$ elements. The multiplicative group of units R^* contains a unique cyclic subgroup $\langle \beta \rangle$ of order $n - 1$. Every element of R can be expressed uniquely as $A + 2B$, where $A, B \in \mathcal{T}$ and

$$\mathcal{T} = \{0, 1, \beta, \dots, \beta^{n-2}\}.$$

Let $\mu : \mathbf{Z}_4 \rightarrow \mathbf{F}_2$ denote the modulo 2 reduction map. We extend μ to R and \mathbf{Z}_4^n in a natural way, and then $\mu(\mathcal{T}) = \mathbf{F}$ and $\mu(\mathbf{Z}_4^n) = \mathbf{F}_2^n$.

DEFINITION 3.1. *Let $m \geq 3$ be odd. The \mathbf{Z}_4 -Goethals code \mathcal{G} of length n is defined by a parity-check matrix*

$$H_{\mathcal{G}} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & \beta & \beta^2 & \dots & \beta^{n-2} \\ 0 & 2 & 2\beta^3 & 2\beta^6 & \dots & 2\beta^{3(n-2)} \end{pmatrix}$$

and the \mathbf{Z}_4 -Preparata code \mathcal{P} by a parity-check matrix $H_{\mathcal{P}}$ which consists of the first two rows of $H_{\mathcal{G}}$.

The codes \mathcal{P} and \mathcal{G} were introduced in the seminal paper [7] by Hammons et al. This paper started the study of \mathbf{Z}_4 -codes by establishing, e.g., that the classical binary nonlinear Preparata and Goethals codes are essentially $\phi(\mathcal{P})$ and $\phi(\mathcal{G})$.

By [9, Lemma 2] a word $(c_X)_{X \in \mathcal{T}}$, with $C_j = \{\mu(X) \mid c_X = j\}$ for $j \in \mathbf{Z}_4$, is a codeword of \mathcal{G} if and only if it satisfies the following equations over \mathbf{F} :

$$(3.1) \quad \begin{aligned} \sum_{x \in \mathbf{F}} c_x &= 0 \quad (\text{in } \mathbf{Z}_4), & \sum_{x \in C_1 \cup C_3} x &= 0, \\ \sum_{\substack{x, y \in C_1 \cup C_3, \\ x < y}} xy &= \sum_{x \in C_2 \cup C_3} x^2, & \sum_{x \in C_1 \cup C_3} x^3 &= 0, \end{aligned}$$

where \leq is some total order on \mathbf{F} .

As we have equations over \mathbf{F} we think from now on that the codewords are indexed with the elements of \mathbf{F} . With the next theorem [7, Theorem 20] we can decrease the needed case-by-case analysis in the proof of the main theorem.

THEOREM 3.2. *The automorphism group of \mathcal{G} contains the doubly transitive group of affine permutations*

$$x \mapsto ax + b, \quad a \in \mathbf{F}^*, b \in \mathbf{F}.$$

3.1. Low-weight codewords and supports. Some of the following facts are well known [7], and the others can be easily deduced from (3.1).

LEMMA 3.3.

- (i) *The minimum Lee distance of \mathcal{G} is 8.*
- (ii) *$\mu(\mathcal{G}) = \mathcal{B}$, the extended double-error-correcting BCH code.*
- (iii) *$\mathcal{G} \cap 2\mathbf{Z}_4^n = 2\mathcal{H}$, the binary extended Hamming code (viewed in \mathbf{Z}_4^n).*
- (iv) *If $\mathbf{c} \in \mathcal{G}$ and $\text{hwe}(\mathbf{c}) = X^i$ with $1 \leq i \leq 6$, then $\text{cwe}(\mathbf{c}) \in \{Y^4, Y^6\}$.*
- (v) *If $\mathbf{c} \in \mathcal{G}$ and $\text{hwe}(\mathbf{c}) = X^7$, then $\text{cwe}(\mathbf{c}) \in \{X^6Y, X^4YZ^2, X^2YZ^4, YZ^6\}$.*
- (vi) *If $\mathbf{c}, \mathbf{d} \in \mathcal{G}$, $\text{hwe}(\mathbf{c}) = \text{hwe}(\mathbf{d}) = X^7$, and $\chi(\mathbf{c}) = \chi(\mathbf{d})$, then $\mathbf{c} = \pm \mathbf{d}$.*
- (vii) *If $\mathbf{c} \in \mathcal{G}$ and $\text{hwe}(\mathbf{c}) = X^8$, then $\text{cwe}(\mathbf{c}) \in \{X^8, X^6Z^2, X^4Z^4, X^2Z^6, Z^8, X^5Y^2Z, X^3Y^2Z^3, XY^2Z^5, Y^8\}$.*
- (viii) *If $\mathbf{c}, \mathbf{d} \in \mathcal{G}$, $\text{swe}(\mathbf{c}) = \text{swe}(\mathbf{d}) = X^6Y^2$, and $\chi(\mathbf{c}) = \chi(\mathbf{d})$, then $\mathbf{c} = \pm \mathbf{d}$.*

(ix) If $\mathbf{c}, \mathbf{d} \in \mathcal{G}$, $\text{swe}(\mathbf{c}) \in \{X^8, Y^8\}$, and $\text{swe}(\mathbf{d}) = X^6Y^2$, then $\chi(\mathbf{c}) \neq \chi(\mathbf{d})$.

The supports of codewords of swe-type X^8 are distributed in a more complicated manner. For example, there are codewords like 11111111 and 11113333 which have the same support. Below we shorten sentences by using the phrase “support of swe-type X^8 ” instead of “support of codeword of swe-type X^8 .”

DEFINITION 3.4. Let S be a subset of the index set \mathbf{F} . The lifting rank of S is $4^{k_1}2^{k_2}$ if a generator matrix of a subcode $\mathcal{G}|_S = \{\mathbf{c} \in \mathcal{G} \mid \chi(\mathbf{c}) \subseteq S\}$ is permutation-equivalent to a matrix of the form

$$G_S = \begin{pmatrix} I_{k_1} & A & B \\ 0 & 2I_{k_2} & 2C \end{pmatrix}.$$

In this paper we consider only the lifting ranks of supports of swe-type X^8 , and hence we always have $k_1 = 1$. We abbreviate by saying that such a support has (lifting) rank k_2 .

The rank counts linearly independent codewords of cwe-type Y^4 within the support. It is easy to see that the rank k satisfies $0 \leq k \leq 3$. The supports of rank 3 are special: they are 3-flats in affine geometry, or, equivalently, minimum weight codewords in the Reed–Muller code $RM(m, m-3)$; see, for example, [16, Chapter 13]. In the next lemma we analyze the possible ranks and the corresponding subcodes.

LEMMA 3.5. Supports of swe-type X^8 divide into the following distinct classes:

- (i) S has rank 3 and $\text{cwe}(\mathcal{G}|_S) = X^8 + 14X^4Z^4 + Z^8 + (W^8 + 14Y^4 + Y^8)$.
- (ii) S has rank 1 and $\text{cwe}(\mathcal{G}|_S) = X^6Z^2 + 2X^4Z^4 + X^2Z^6 + (W^8 + 2Y^4 + Y^8)$.
- (iii) S has rank 0 and $\text{cwe}(\mathcal{G}|_S) = X^6Z^2 + X^2Z^6 + (W^8 + Y^8)$.
- (iv) S has rank 0 and $\text{cwe}(\mathcal{G}|_S) = X^8 + Z^8 + (W^8 + Y^8)$.
- (v) S has rank 0 and $\text{cwe}(\mathcal{G}|_S) = 2X^4Z^4 + (W^8 + Y^8)$.

Proof. If S has rank 3, the cwe of the subcode follows directly.

If $S = \{x_1, x_2, \dots, x_8\}$ has rank ≥ 1 but is not a 3-flat, there is a 2-flat, say, $\{x_1, x_2, x_3, x_4\}$ in S and we can extend it to a 3-flat $\{x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3\}$, $y_i \notin S$. This support is in the class (i) and considering case-by-case we see that only the combination (ii) can exist. In this case codewords 11131113 and 11133331 have the same support; that is, in a codeword of cwe-type X^6Z^2 both 2-flats include one 3-position. All other possibilities contradict the Lee minimum distance of \mathcal{G} .

The classes (iii)–(v) clearly list all types of supports of rank 0. \square

When $m = 3$ the only support S of size 8 is in the class (i), and $\mathcal{G} = \mathcal{G}|_S$ has 32 codewords. In section 5 we will see that when $m = 5$ only the classes (i) and (ii) are nonempty. By computer we have checked that when $m \in \{7, 9\}$ all the classes are nonempty, and it is reasonable to expect that this is true also when $m \geq 11$.

Another way to classify the supports is to think of the relations between the binary codes \mathcal{H} and \mathcal{B} and the \mathbf{Z}_4 -code \mathcal{G} . Let $\mathbf{c} \in \mathcal{H}$ be a codeword (or support) of hwe-type X^8 . There are two possibilities: \mathbf{c} belongs to the subcode \mathcal{B} , or not. If $\mathbf{c} \in \mathcal{B}$ there are again two possibilities: \mathbf{c} can be lifted to \mathcal{G} such that changing 1’s in some positions to 3’s gives a codeword from \mathcal{G} , or not. With a little abuse of notation we can state that these supports divide into the following three distinct classes:

- (A) $\mathbf{c} \in \mathcal{H} \setminus \mathcal{B}$,
- (B) $\mathbf{c} \in \mathcal{B} \setminus \mathcal{G}$,
- (C) $\mathbf{c} \in \mathcal{G}$.

The classes (i)–(v) together form the class (C), and the classes (A)–(B) contain all supports of cwe-type Y^8 in \mathcal{G} which are not in (C). We denote the supports of swe-type X^6Y^2 as class (D), and by (ix) in Lemma 3.3 it is distinct from (A)–(C). Therefore all supports of size 8 in \mathcal{G} divide into four distinct classes (A)–(D).

4. Known 3-designs. In this section we describe some known 3-designs which are connected to our new results. For simplicity we assume m to be odd ($v = n = 2^m$).

The Assmus–Mattson theorem [1] gives a sufficient condition for the codewords of a fixed Hamming weight in a linear code over a finite field to form a t -design. This theorem is applicable to the codes \mathcal{H} and \mathcal{B} , i.e., $\mathcal{G} \cap 2\mathbf{Z}_4^n$ and $\mu(\mathcal{G})$, with well-known weight distributions. The parameter λ and the number of blocks, say b , are linked with the identity $\binom{n}{t}\lambda = \binom{k}{t}b$, and we can calculate the following λ 's.

THEOREM 4.1.

- (i) *The supports of size 4 in \mathcal{H} form a $3-(n, 4, 1)$ design.*
- (ii) *The supports of size 6 in \mathcal{H} form a $3-(n, 6, \frac{(n-4)(n-8)}{6})$ design.*
- (iii) *The classes (A)–(C) form a $3-(n, 8, \frac{(n-4)(n-6)(n^2-15n+71)}{120})$ design.*
- (iv) *The supports of size 6 in \mathcal{B} form a $3-(n, 6, \frac{n-8}{6})$ design.*
- (v) *The classes (B)–(C) form a $3-(n, 8, \frac{n^3-25n^2+246n-760}{120})$ design.*

The following theorem can be proved simply by counting the 3-flats which contain some fixed three points.

THEOREM 4.2. *The class (i) defines a $3-(n, 8, \frac{n-4}{4})$ design.*

Next we recall two classical related results [17, 2] in the form of one theorem. A binary distance invariant (for the definition, see [16, page 40]) code is called Preparata-like or Goethals-like if it has the same weight distribution as $\phi(\mathcal{P})$ or $\phi(\mathcal{G})$, respectively.

THEOREM 4.3. *The supports of fixed size in any Preparata-like code form a 3-design. The supports of fixed size in any Goethals-like code form a 3-design.*

From this theorem we derive one 3-design needed in section 7. By [7] we can count the number of codewords of Hamming weight 8 in $\phi(\mathcal{G})$ and hence the corresponding λ . This parameter can be derived also from [2, Theorem 3] or more directly from [14, Proposition 1]: the expression for $\mu_{3,5}$ is equal to λ .

COROLLARY 4.4. *The supports of size 8 in the Goethals-like code $\phi(\mathcal{G})$ form a $3-(2n, 8, \frac{(2n-4)(4n-17)}{60})$ design.*

The next result by Shin, Kumar, and Helleseth [18] is the starting point for this paper. It gives the first “nonbinary” family of 3-designs from the \mathbf{Z}_4 -Goethals codes.

THEOREM 4.5. *The supports of size 7 in \mathcal{G} form a $3-(n, 7, \frac{14}{3}(n-8))$ design.*

In [19] the Assmus–Mattson type theorem is introduced and used together with the previous theorem to derive the following.

THEOREM 4.6. *The codewords of any fixed hwe-type in \mathcal{G} form a 3-design possibly with repeated blocks.*

5. New 3-designs. We now state our main theorem and deduce some corollaries. All the design families in this section are new in the sense that they are not listed in [15, Table 3.31]: the known infinite families of simple t -designs with $t \geq 3$.

THEOREM 5.1. *The class (ii) defines a $3-(n, 8, \frac{14}{3}(n-8))$ design.*

The proofs of Theorem 5.1 and Corollary 5.2 are postponed to sections 6 and 7.

COROLLARY 5.2. *The class (C) forms a $3-(n, 8, \frac{32n^2-985n+5892}{60})$ design.*

COROLLARY 5.3. *The class (B) defines a $3-(n, 8, \frac{(n-8)(n-32)(n-49)}{120})$ design.*

Proof. Subtract λ in Corollary 5.2 from λ in (v) in Theorem 4.1. \square

COROLLARY 5.4. *The class (D) forms a $3-(n, 8, \frac{56}{15}(n-8)(n-12))$ design.*

Proof. By Theorem 4.6 the codewords of hwe-type X^8 define a 3-design with repeated blocks. We can count its parameter $\lambda^* = (n^4 - 25n^3 + 1269n^2 - 21390n +$

100648)/120 from $\text{cwe}(\mathcal{G}^\perp)$ [19]. This design contains codewords of swe-types Y^8 , X^8 , and X^6Y^2 by Lemma 3.3. We drop out all codewords of swe-types Y^8 and X^8 , and by (iii) in Theorem 4.1 and λ' in (7.1) we get a 3-design with λ equal to

$$\lambda^* - \lambda_{4.1(iii)} - \lambda' = \frac{112}{15}(n - 8)(n - 12).$$

We take the supports of the remaining codewords, and by item (viii) in Lemma 3.3 we get a simple 3-design with a parameter $\lambda/2$. \square

COROLLARY 5.5. *The supports of size 8 in \mathcal{G} form a 3- $(n, 8, \lambda)$ design with*

$$\lambda = \frac{n^4 - 25n^3 + 693n^2 - 10030n + 44712}{120}.$$

Proof. Add λ in (iii) in Theorem 4.1 to λ in Corollary 5.4. \square

Example 5.6 (3-designs with $n = 32$). The designs of length 32 are quite different from the longer ones. By counting the parameters of all designs above we see that the classes (iii)–(v) and (B) are empty. All in all we get only three new designs:

- 3-(32, 8, 112) design (class (ii)),
- 3-(32, 8, 1792) design (class (D)),
- 3-(32, 8, 5523) design (all supports of size 8).

Duursma et al. [5] noticed that for the length 32 the automorphism group of \mathcal{G} is 3-homogeneous (any 3-subset can be mapped to an arbitrary 3-subset), and hence codewords of any fixed cwe-type form a 3-design possibly with repeated blocks. Applying this result to the second design we can split it in 3-(32, 8, 672) and 3-(32, 8, 1120) designs corresponding to the cwe-types X^5Y^2Z and $X^3Y^2Z^3$, respectively.

Interestingly, for length 32 we have 3-(32, 6, 112), 3-(32, 7, 112), and 3-(32, 8, 112) designs by (ii) in Theorem 4.1, Theorem 4.5, and Theorem 5.1, respectively.

Example 5.7 (3-designs with $n = 128$). Now all classes (i)–(v) and (A)–(D) are nonempty and we get all five new designs:

- 3-(128, 8, 560) design (class (ii)),
- 3-(128, 8, 6735) design (class (C)),
- 3-(128, 8, 7584) design (class (B)),
- 3-(128, 8, 51968) design (class (D)),
- 3-(128, 8, 1884347) design (all supports of size 8).

By computer we can verify also the existence of the following designs:

- 3-(128, 8, 2688) design (class (iii)),
- 3-(128, 8, 3456) design (classes (iv) and (v)).

By computer we also claim that the class (iv) alone does not define a 3-design.

Example 5.8 (3-designs with $n = 512$). We do not list the parameters of the five new designs anymore but remark that, again, by computer we get

- 3-(512, 8, 56448) design (class (iii)),
- 3-(512, 8, 72576) design (classes (iv) and (v)).

CONJECTURE 5.9. *The class (iii) forms a 3-design.*

6. Proof of Theorem 5.1. To prove the main theorem we have to show that any three distinct coordinate positions are included in equally many supports of cwe-type X^6Z^2 and rank 1. By Theorem 3.2 we can assume that these positions are 0, 1, and an arbitrary element $a \in \mathbf{F}^{**}$. In this section all supports and codewords are assumed to be of cwe-type X^6Z^2 and rank 1 unless otherwise stated.

Lemma 3.5 shows that codewords (of this considered type) can be identified with their supports, and we know that the support is a union of two 2-flats and both of

TABLE 1
All combinations of three coordinates.

Case	1	1	1	3	1	1	1	3	Frequency
	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	
(0a)	0	1	a						$(n-8)/6$
(0b)	0	1	a						$(n-8)/6$
(0b')	0	a	1						$(n-8)/6$
(0b'')	1	a	0						$(n-8)/6$
(1a)	0	1			a				$\frac{2(n-8)}{3}$
(1b)	0	1					a		
(2a)	0		1		a				$\frac{n-8}{2}$
(2b)	1		0		a				
(3a)	0		1				a		$\frac{n-8}{6}$
(3b)	1		0				a		
(1')	0	a			1			1	$\frac{2(n-8)}{3}$
(2')	0	a			1			1	
(3')	0		a					1	$\frac{n-8}{2}$
(1'')	0		a					1	
(2'')	1	a			0			0	$\frac{2(n-8)}{3}$
(3'')	1	a			0			0	
(1''')	1		a		0			0	$\frac{n-8}{2}$
(2''')	1		a		0			0	
(3''')	1		a				0	0	$\frac{n-8}{6}$
(1''''')	1		a				0	0	

them contain one 3-position. In other words, codeword is split in the two 2-flats as 1113 and 1113. This leads to a total of 22 combinations of positions 0, 1, and a among the two 2-flats and 1's and 3's as shown in Table 1. The number of codewords belonging to each combination with a fixed a is also shown.

Next we verify the different frequencies and by summing them up we see that λ is equal to $14(n - 8)/3$ and the supports, indeed, form a 3-design.

Assume that we have a codeword which is counted in case (0b); that is, the corresponding support includes the positions 0, 1, and a . Using permutation $x \mapsto x/a$ and Theorem 3.2 we get a codeword which includes the positions 0, $1/a$, and 1. This permuted codeword is counted in case (0b') with a parameter $1/a \in \mathbf{F}^{**}$. Conversely, the codewords counted in case (0b) are permuted versions of codewords in case (0b'), and therefore we need to prove the frequency only for one of them.

With the same permutation we can link cases (1a)–(3b) with (1')–(3'). Another permutation $x \mapsto x + 1$ links case (0b') with (0b'') and cases (1')–(3') with (1'')–(3''). We conclude that it suffices to prove only cases (0a), (0b), and (1a)–(3b).

6.1. Syndrome equations. Next we consider the equations which the support $\{x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4\}$ from Table 1 should satisfy. The sets $\{x_1, x_2, x_3, x_4\}$ and $\{y_1, y_2, y_3, y_4\}$ form the two 2-flats and 3's are thought to be in the positions x_4 and y_4 . By (3.1) and the 2-flat structure the following equations should hold:

$$\begin{aligned} \sigma_1(x_1, x_2, x_3, x_4) &= 0, \\ \sigma_1(y_1, y_2, y_3, y_4) &= 0, \\ \sigma_2(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) &= x_4^2 + y_4^2, \\ S_3(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4) &= 0, \end{aligned}$$

where $\sigma_k(a_1, \dots, a_n) = \sum_{1 \leq i_1 < \dots < i_k \leq n} a_{i_1} \dots a_{i_k}$ is the k th elementary symmetric polynomial and $S_k(a_1, \dots, a_n) = \sum_{i=1}^n a_i^k$ is the sum of k th powers.

We abbreviate the regularly used terms $\sigma(x_1, x_2, x_3, x_4)$ and $\sigma(x_1, x_2, x_3)$ to $\sigma(x_i)$ and $\sigma'(x_i)$, respectively. Similar notations hold for S_3 's and y_i 's.

We simplify the third equation with the first two equations:

$$\sigma'_2(x_i) + \sigma'_2(y_i) = \sigma_1(x_i)\sigma_1(y_i) + \sigma'_1(x_i)x_4 + x_4^2 + \sigma'_1(y_i)y_4 + y_4^2 = 0.$$

With Newton's identity $S_3 = \sigma_1^3 + \sigma_1\sigma_2 + \sigma_3$ the fourth equation becomes

$$\begin{aligned} S_3(x_i) + S_3(y_i) &= \sigma_1(x_i)^3 + \sigma_1(x_i)\sigma_2(x_i) + \sigma_3(x_i) \\ &+ \sigma_1(y_i)^3 + \sigma_1(y_i)\sigma_2(y_i) + \sigma_3(y_i) = \sigma_3(x_i) + \sigma_3(y_i) = 0. \end{aligned}$$

All in all the considered support should satisfy

$$(6.1) \quad \begin{aligned} \sigma_1(x_i) &= \sigma_1(y_i) = 0, \\ \sigma'_2(x_i) &= \sigma'_2(y_i), \\ \sigma_3(x_i) &= \sigma_3(y_i). \end{aligned}$$

If the variables x_i and y_i are distinct, the corresponding codeword is of the desired type and rank. The items (iv)–(vi) from Lemma 3.3 imply easily that the only possible overlapping of the variables x_i and y_i satisfying (6.1) is the case where the 2-flats are equal and $x_4 = y_4$; that is, they form the support $\{0, 1, a, a + 1\}$ of cwe-type Y^4 . So the solutions of (6.1) have one extra codeword which must be excluded.

6.2. Cases (0a) and (0b). In case (0a) we set $x_1 = 0, x_2 = 1,$ and $x_3 = a.$ By (6.1) we have $x_4 = a + 1,$ and therefore $\sigma'_2(x_i) = a$ and $\sigma_3(x_i) = a^2 + a.$ We have four unknown variables y_i which satisfy by (6.1)

$$\begin{aligned} \sigma'_1(y_i) &= y_4, \\ \sigma'_2(y_i) &= a, \\ \sigma_3(y_i) &= \sigma'_2(y_i)y_4 + \sigma'_3(y_i) = a\sigma'_1(y_i) + \sigma'_3(y_i) = a^2 + a. \end{aligned}$$

We think of $\sigma'_1(y_i) = \sigma$ as a free variable, and the above equations show that $y_1, y_2,$ and y_3 are zeros of the polynomial

$$p(T) = T^3 + \sigma'_1(y_i)T^2 + \sigma'_2(y_i)T + \sigma'_3(y_i) = T^3 + \sigma T^2 + aT + a\sigma + a^2 + a.$$

Clearly, interchanging the roles of the variables $y_1, y_2,$ and y_3 affects neither the above polynomial nor the corresponding codeword. Therefore the polynomials $p(T)$ that have three distinct zeros in \mathbf{F} correspond to the codewords in case (0a) or possibly to the extra codeword.

The polynomial $p(T)$ always has three distinct zeros in its splitting field as $p(T)$ and its derivative $p'(T) = T^2 + a$ have no common zeros. The question is, For how many values of σ all zeros of $p(T)$ are in \mathbf{F} ?

We substitute $T = U + \sigma$ and $p(T)$ transforms to $U^3 + (\sigma^2 + a)U + a^2 + a.$ For $\sigma = \sqrt{a}$ this polynomial has only one zero in $\mathbf{F},$ and we can ignore that case. We substitute again with $U = (\sigma + \sqrt{a})V$ and divide by $(\sigma + \sqrt{a})^3$ and get

$$V^3 + V + \frac{a^2 + a}{(\sigma + \sqrt{a})^3}.$$

By Lemma 2.2 this polynomial has three zeros in \mathbf{F} for $(n - 2)/6$ values of σ since $a^2 + a \neq 0$ and $x \mapsto x^3$ is a bijection $\mathbf{F}^* \rightarrow \mathbf{F}^*$. Our substitutions preserve the number of zeros, and hence $p(T)$ has three zeros in \mathbf{F} for $(n - 2)/6$ values of σ .

The codeword of cwe-type Y^4 would have $\{y_1, y_2, y_3\} = \{0, 1, a\}$, $\sigma = a + 1$, and $p(T) = T^3 + (a + 1)T^2 + aT$. Excluding this extra codeword we have all in all $(n - 8)/6$ codewords in case (0a) as claimed in Table 1.

In case (0b) we have $x_1 = 0, x_2 = 1, x_4 = a$, and $x_3 = a + 1$. The automorphism $x \mapsto x + 1$ links case (0b) with (0a), and the frequencies are the same.

6.3. Cases (1a) and (1b). We have $x_1 = 0, x_2 = 1$, and $y_1 = a$ in case (1a). Now there are unknown variables in both 2-flats, and the calculations are more complicated. We denote the variable x_3 by x and the elementary symmetric polynomials of y_2 and y_3 by $\sigma_1 = y_2 + y_3$ and $\sigma_2 = y_2y_3$. The variables y_2 and y_3 are zeros of the polynomial $T^2 + \sigma_1T + \sigma_2$, and by Lemma 2.1 the condition $\text{Tr}(\sigma_2/\sigma_1^2) = 0$ must hold.

The extra codeword of cwe-type Y^4 has the support $\{0, 1, a, a + 1\}$, and then $\sigma_1 \in \{1, a, a + 1\}$. On the other hand, if $\sigma_1 \in \{0, 1, a, a + 1\}$, then the 2-flats would overlap or the support would have rank 3. As rank 3 contradicts (6.1), the only possible codeword is of cwe-type Y^4 . To exclude this extra word we can restrict ourselves to the cases $\sigma_1 \in \mathbf{F}^a = \mathbf{F} \setminus \{0, 1, a, a + 1\}$. We make this same restriction also in the forthcoming subsections.

By (6.1) we know that $x_4 = x + 1, y_4 = a + \sigma_1, x = a\sigma_1 + \sigma_2$, and

$$(6.2) \quad x^2 + x = a\sigma_2 + a\sigma_1(a + \sigma_1) + \sigma_2(a + \sigma_1) = \sigma_1\sigma_2 + a^2\sigma_1 + a\sigma_1^2.$$

We substitute the value of x from the third equation to (6.2) and get a quadratic equation in the unknown σ_2 :

$$(6.3) \quad \sigma_2^2 + (\sigma_1 + 1)\sigma_2 = (a^2 + a)\sigma_1(\sigma_1 + 1).$$

As σ_2 is an element of \mathbf{F} , again by Lemma 2.1, the condition

$$(6.4) \quad \text{Tr}\left(\frac{(a^2 + a)\sigma_1}{\sigma_1 + 1}\right) = \text{Tr}\left(\frac{(a^2 + a)\sigma_1}{\sigma_1 + 1} + a^2 + a\right) = \text{Tr}\left(\frac{a^2 + a}{\sigma_1 + 1}\right) = 0$$

must hold. We simplified the condition using the identity $\text{Tr}(a^2) = \text{Tr}(a)$.

By dividing (6.3) by σ_1^2 from both sides, the other trace condition takes the form

$$\text{Tr}\left(\frac{\sigma_2}{\sigma_1}\right) = \text{Tr}\left(\frac{\sigma_2^2}{\sigma_1^2} + \frac{\sigma_2}{\sigma_1}\right) + \text{Tr}(a^2 + a) + \text{Tr}\left(\frac{a^2 + a}{\sigma_1}\right) = \text{Tr}\left(\frac{a^2 + a}{\sigma_1}\right) = 0.$$

Therefore we should count the number

$$\begin{aligned} N_a &= \left| \left\{ \sigma_1 \in \mathbf{F}^a \mid \text{Tr}\left(\frac{a^2 + a}{\sigma_1}\right) = 0 \text{ and } \text{Tr}\left(\frac{a^2 + a}{\sigma_1 + 1}\right) = 0 \right\} \right| \\ &= \frac{1}{4} \sum_{i,j=0}^1 \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{i \cdot \text{Tr}\left(\frac{a^2 + a}{\sigma_1}\right) + j \cdot \text{Tr}\left(\frac{a^2 + a}{\sigma_1 + 1}\right)} \\ &= \frac{1}{4} (N_{0,0} + N_{0,1} + N_{1,0} + N_{1,1}), \end{aligned}$$

where $N_{i,j}$ is the inner sum in the second line.

The number of codewords in case (1a) is twice the number N_a . This can be seen as follows: for every σ_1 which satisfies the trace conditions we have two solutions, σ_2 and $\sigma_2 + \sigma_1 + 1$, for (6.3). The roles of y_2 and y_3 can be changed without affecting the codeword, and hence σ_2 corresponds to one codeword. The other solution $\sigma_2 + \sigma_1 + 1$ gives a different codeword since $\sigma_1 \neq 1$. It also satisfies the trace condition $\text{Tr}((\sigma_2 + \sigma_1 + 1)/\sigma_1^2) = \text{Tr}(\sigma_2/\sigma_1^2) = 0$ by the identity $\text{Tr}((a + 1)/a^2) = 0$.

Clearly, $N_{0,0} = n - 4$. In the calculation of $N_{0,1}$ we use the substitution $z = (a^2 + a)/(\sigma_1 + 1)$:

$$\begin{aligned} N_{0,1} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1+1}\right)} = \sum_{z \in \mathbf{F} \setminus \{a^2+a, 0, a, a+1\}} (-1)^{\text{Tr}(z)} \\ &= -(-1)^{\text{Tr}(a^2+a)} - (-1)^{\text{Tr}(0)} - (-1)^{\text{Tr}(a)} - (-1)^{\text{Tr}(a+1)} = -2. \end{aligned}$$

By using the substitution $z = (a^2 + a)/\sigma_1$

$$N_{1,0} = \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1}\right)} = \sum_{z \in \mathbf{F} \setminus \{0, a^2+a, a+1, a\}} (-1)^{\text{Tr}(z)} = -2.$$

By the substitution $z = 1/\sigma_1$ we get

$$N_{1,1} = \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1} + \frac{a^2+a}{\sigma_1+1}\right)} = \sum_{z \in \mathbf{F} \setminus \{0, 1, \frac{1}{a}, \frac{1}{a+1}\}} (-1)^{\text{Tr}\left((a^2+a)z + \frac{(a^2+a)z}{z+1}\right)}.$$

As in (6.4) we drop z from the last numerator and substitute $u = z + 1$ and $w = (a^2 + a)u$:

$$\begin{aligned} N_{1,1} &= \sum_{u \in \mathbf{F} \setminus \{1, 0, \frac{a+1}{a}, \frac{a}{a+1}\}} (-1)^{\text{Tr}\left((a^2+a)u + a^2+a + \frac{a^2+a}{u}\right)} \\ &= \sum_{w \in \mathbf{F} \setminus \{a^2+a, 0, a^2+1, a^2\}} (-1)^{\text{Tr}\left(w + \frac{(a^2+a)^2}{w}\right)} = K(a^2 + a) + 1 \end{aligned}$$

since clearly $K((a^2 + a)^2) = K(a^2 + a)$. By combining the above results we have

$$N_a = \frac{1}{4}(n - 7 + K(a^2 + a)).$$

Case (1b) goes almost as above. Now $y_4 = a$, $y_1 = a + \sigma_1$, $x = \sigma_1^2 + a\sigma_1 + \sigma_2$, (6.2) still holds, but instead of (6.3) we get

$$\sigma_2^2 + (\sigma_1 + 1)\sigma_2 = (a^2 + a)\sigma_1(\sigma_1 + 1) + \sigma_1^2(\sigma_1 + 1)^2.$$

Therefore the task is to calculate the number

$$N'_a = \left| \left\{ \sigma_1 \in \mathbf{F}^a \mid \text{Tr}\left(\frac{a^2+a}{\sigma_1} + \sigma_1 + 1\right) = 0 \text{ and } \text{Tr}\left(\frac{a^2+a}{\sigma_1+1} + \sigma_1\right) = 0 \right\} \right|.$$

For every σ_1 counted in N'_a we get two codewords as in case (1a), but this time every codeword is counted three times in the number $2N'_a$. This follows from the observation that among the three coordinates y_1 , y_2 , and y_3 we can choose two coordinates with

three ways. The two chosen are identified with σ_1 and σ_2 , and the third one is equal to $\sigma_1 + a$. Therefore the number of codewords in case (1b) is equal to $2N'_a/3$.

As above we know that $N'_{0,0} = n - 4$. With the substitution $z = \sigma_1 + 1$ we have

$$\begin{aligned}
 N'_{0,1} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1+1} + \sigma_1\right)} = \sum_{z \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{z} + z + 1\right)} = -K(a^2 + a) - 3, \\
 N'_{1,0} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1} + \sigma_1 + 1\right)} = -K(a^2 + a) - 3, \\
 N'_{1,1} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1} + \frac{a^2+a}{\sigma_1+1} + 1\right)} = -N_{1,1} = -K(a^2 + a) - 1.
 \end{aligned}$$

Then $N'_a = (n - 11 - 3K(a^2 + a))/4$, and the total number of codewords in cases (1a) and (1b) together is equal to the number in Table 1:

$$2N_a + \frac{2}{3}N'_a = \frac{n - 7 + K(a^2 + a)}{2} + \frac{n - 11 - 3K(a^2 + a)}{6} = \frac{2(n - 8)}{3}.$$

6.4. Cases (2a) and (2b). The situation in case (2a) is as follows: $x_1 = 0$, $x_4 = 1$, and $y_1 = a$. As above we denote $x = x_3$, $\sigma_1 = y_2 + y_3$, and $\sigma_2 = y_2y_3$. We have by (6.1) that $x_2 = x + 1$, $y_4 = a + \sigma_1$, and (6.2) holds, but this time the equation for σ'_2 gives

$$x^2 + x = a\sigma_1 + \sigma_2.$$

By combining this with (6.2) we get $\sigma_2 = \sigma_1(a\sigma_1 + a^2 + a)/(\sigma_1 + 1)$. As we are interested in roots x which are in \mathbf{F} the variable σ_1 should satisfy

$$(6.5) \quad \text{Tr}\left(a\sigma_1 + \frac{\sigma_1(a\sigma_1 + a^2 + a)}{\sigma_1 + 1}\right) = \text{Tr}\left(\frac{a^2\sigma_1}{\sigma_1 + 1}\right) = \text{Tr}\left(\frac{a^2}{\sigma_1 + 1} + a^2\right) = 0$$

by Lemma 2.1. On the other hand, σ_1 should also satisfy

$$\begin{aligned}
 (6.6) \quad \text{Tr}\left(\frac{\sigma_2}{\sigma_1^2}\right) &= \text{Tr}\left(\frac{a\sigma_1 + a^2 + a}{\sigma_1(\sigma_1 + 1)}\right) + \text{Tr}\left(\frac{a^2\sigma_1}{\sigma_1 + 1}\right) \\
 &= \text{Tr}\left(\frac{(a^2 + a)(\sigma_1 + 1)}{\sigma_1(\sigma_1 + 1)} + \frac{a^2\sigma_1(\sigma_1 + 1)}{\sigma_1(\sigma_1 + 1)}\right) \\
 &= \text{Tr}\left(\frac{a^2 + a}{\sigma_1} + a^2\right) = 0.
 \end{aligned}$$

In the first line we added a term which is equal to zero by (6.5).

This time we are interested in the number

$$M_a = \left| \left\{ \sigma_1 \in \mathbf{F}^a \mid \text{Tr}\left(\frac{a^2+a}{\sigma_1} + a\right) = 0 \text{ and } \text{Tr}\left(\frac{a^2}{\sigma_1+1} + a\right) = 0 \right\} \right|$$

which is the number of codewords in case (2a): for every σ_1 satisfying the trace conditions there is one σ_2 , and with them we get solutions x and $x + 1$ for the equation (6.2), but they correspond to the same codeword.

We count the number M_a as above. Clearly, $M_{0,0} = n - 4$ and

$$\begin{aligned} M_{0,1} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2}{\sigma_1+1}+a\right)} = \sum_{z \in \mathbf{F} \setminus \{a^2+a, a, \frac{a}{a+1}, 0\}} (-1)^{\text{Tr}(z)} \\ &= -2 - (-1)^{\text{Tr}(a)} - (-1)^{\text{Tr}\left(\frac{a}{a+1}\right)}, \\ M_{1,0} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1}+a\right)} = (-1)^{\text{Tr}(a)} N_{1,0} = -2(-1)^{\text{Tr}(a)}. \end{aligned}$$

We use the substitutions $z = 1/\sigma_1$, $u = z + 1$, and $w = (a^2 + a)u$, and then

$$\begin{aligned} M_{1,1} &= \sum_{\sigma_1 \in \mathbf{F}^a} (-1)^{\text{Tr}\left(\frac{a^2+a}{\sigma_1} + \frac{a^2}{\sigma_1+1}\right)} = \sum_{z \in \mathbf{F} \setminus \{0, 1, \frac{1}{a}, \frac{1}{a+1}\}} (-1)^{\text{Tr}\left((a^2+a)z + \frac{a^2}{z+1} + a^2\right)} \\ &= \sum_{u \in \mathbf{F} \setminus \{0, 1, \frac{a+1}{a}, \frac{a}{a+1}\}} (-1)^{\text{Tr}\left((a^2+a)u + \frac{a^2}{u} + a\right)} \\ &= \sum_{w \in \mathbf{F} \setminus \{0, a^2+a, a^2+1, a^2\}} (-1)^{\text{Tr}\left(w + \frac{a^3(a+1)}{w} + a\right)} \\ &= (-1)^{\text{Tr}(a)} K(a^3(a+1)) - 2 - (-1)^{\text{Tr}\left(\frac{1}{a+1}\right)}. \end{aligned}$$

We conclude that

$$M_a = \frac{1}{4} \left(n - 8 - 3(-1)^{\text{Tr}(a)} + (-1)^{\text{Tr}(a)} K(a^3(a+1)) \right).$$

Case (2b) is linked with case (2a) by the automorphism $x \mapsto x + 1$. Therefore the total number of codewords in cases (2a) and (2b) is equal to

$$M_a + M_{a+1} = \frac{2n - 16 + (-1)^{\text{Tr}(a)} \left(K(a^3(a+1)) - K((a+1)^3 a) \right)}{4} = \frac{n - 8}{2}$$

by Theorem 2.4. This is exactly the frequency given in Table 1.

6.5. Cases (3a) and (3b). We have the following dependencies in case (3a): $x_1 = 0$, $x_4 = 1$, and $y_4 = a$. As before we denote $x = x_3$, $\sigma_1 = y_2 + y_3$, and $\sigma_2 = y_2 y_3$. By (6.1) we know that $x_2 = x + 1$, $y_1 = a + \sigma_1$, (6.2) holds, and $x^2 + x = \sigma_1^2 + a\sigma_1 + \sigma_2$. As in the previous cases we can solve $\sigma_2 = (a + 1)\sigma_1(\sigma_1 + a)/(\sigma_1 + 1)$, and the trace conditions get forms

$$\text{Tr}\left(\frac{\sigma_2}{\sigma_1^2}\right) = \text{Tr}\left(\frac{(a+1)(\sigma_1+a)}{\sigma_1(\sigma_1+1)}\right) = 0$$

and

$$\begin{aligned} \text{Tr}(\sigma_1^2 + a\sigma_1 + \sigma_2) &= \text{Tr}\left(\sigma_1^2 + a\sigma_1 + \frac{(a+1)\sigma_1(\sigma_1+a)}{\sigma_1+1}\right) = \text{Tr}\left(\frac{\sigma_1^3 + a^2\sigma_1}{\sigma_1+1}\right) \\ &= \text{Tr}\left(\frac{\sigma_1^3 + a^2\sigma_1}{\sigma_1+1} + \sigma_1^2 + \sigma_1\right) = \text{Tr}\left(\frac{(a+1)^2\sigma_1}{\sigma_1+1}\right) = 0. \end{aligned}$$

We claim that the number of solutions to these two mutual trace equations is equal to M_{a+1} : the substitution $a \mapsto a + 1$ transforms the middle term in (6.6) to the

TABLE 2
Half of the eight left/right combinations.

Case	LLL	LLR	LRL	RLL	Frequency
(a)	222 2	222 2	222 2	222 2	1
(b)	111 1112	113 1132	131 1132	311 1132	$\frac{2(n-8)}{3}$
	111 1332	113 3332	131 3332	311 3332	
(c)	211 1111	213 1113	231 1113	211 1111	$\frac{n-8}{3}$
	211 1133	213 1333	231 1333	211 1133	
	211 3333			211 3333	
(c)	121 1111	123 1113	121 1111	321 1113	$\frac{n-8}{3}$
	121 1133	123 1333	121 1133	321 1333	
	121 3333		121 3333		
(c)	112 1111	112 1111	132 1113	312 1113	$\frac{n-8}{3}$
	112 1133	112 1133	132 1333	312 1333	
	112 3333	112 3333			
(d)	111 11111	113 11113	131 11113	311 11113	?
	111 11133	113 11333	131 11333	311 11333	
	111 13333	113 33333	131 33333	311 33333	
Total	λ_b	λ_b	λ_b	λ_b	λ_b

former equation and the middle expression in (6.5) to the latter equation. Like in case (1b) every codeword is calculated three times, and hence the number of codewords in case (3a) is equal to $M_{a+1}/3$.

Cases (3a) and (3b) are connected via the automorphism $x \mapsto x + 1$, and thus the number of codewords in case (3b) is equal to $M_a/3$. The total number of codewords in cases (3a) and (3b) is then

$$\frac{M_{a+1} + M_a}{3} = \frac{n - 8}{6}$$

as claimed in Table 1. This concludes our proof for the main theorem.

7. Proof of Corollary 5.2. We prove that the number of supports of swe-type X^8 that include three fixed coordinates does not depend on the coordinates.

Let us consider some consequences of Definition 2.6 and Corollary 4.4. If $\mathbf{c} \in \mathcal{G}$ is some codeword with three fixed coordinates, we can choose the corresponding positions of $\phi(\mathbf{c})$ from either \mathbf{c}_L or \mathbf{c}_R . These eight left/right combinations of “binary” positions give us sets of codewords which we can count.

Take, for example, all three positions from the left side. The codewords of $\phi(\mathcal{G})$ that include these “binary” positions are images of \mathbf{Z}_4 -codewords that have either 1 or 2 in the three coordinates. If some position is chosen from the right side, then the corresponding \mathbf{Z}_4 -codeword should have either 2 or 3 in this coordinate. We illustrate half of the combinations in Table 2. The remaining four combinations are obtained by multiplying the whole table by -1 .

By Corollary 4.4 there are $\lambda_b = (2n - 4)(4n - 17)/60$ supports $\phi(\mathbf{c})$ of size 8 in $\phi(\mathcal{G})$ containing three “binary” positions. In Table 2 we list all possible cwe-types of the corresponding \mathbf{Z}_4 -codewords \mathbf{c} of Lee weight 8. The frequency column gives the number of codewords in *one* column.

In case (a) the frequencies come from (i) in Theorem 4.1 and in cases (b) and (c) from Table 3 or [18]. We conclude that the codewords in case (d) in all left/right combinations, that is, the codewords of swe-type X^8 , form a 3-design with repeated

blocks and

$$(7.1) \quad \lambda' = 8? = 8 \left(\frac{(2n-4)(4n-17)}{60} - \frac{5(n-8)}{3} - 1 \right) = \frac{4(4n^2 - 75n + 404)}{15}.$$

We make this design simple using Lemma 3.5 and Theorems 4.2 and 5.1. For every codeword \mathbf{c} of swe-type X^8 there is a codeword $-\mathbf{c}$ with the same support, and it can be excluded. This settles the supports of rank 0. For the supports of rank 3 and 1 there are still 7 and 1 extra codewords, respectively, and therefore the simple design has

$$\lambda = \frac{\lambda'}{2} - 7\lambda_{4.2} - \lambda_{5.1} = \frac{32n^2 - 985n + 5892}{60}.$$

8. Equivalence of Theorems 4.5 and 5.1. There is a considerable similarity between the proofs of Theorems 4.5 and 5.1: the calculations involve similar exponential sums and Kloosterman sum identities, and also λ 's are equal. This suggests that there might be a relation between the blocks of these designs. Indeed, a strong structural connection is illustrated in Table 3 and explained below. This link makes Theorems 4.5 and 5.1 equivalent, and section 6 gives us a simpler and more uniform proof for Theorem 4.5 than the original one [18].

We assume that we have three fixed coordinates and consider blocks containing them. The corresponding \mathbf{Z}_4 -symbols in these positions are shown in the Fix column. Every block of size 8 is viewed as a codeword of cwe-type X^6Z^2 and rank 1, but the blocks of size 7 are viewed as codewords of cwe-types X^6Y , X^4YZ^2 , and X^2YZ^4 , depending on the situation. The case notations refer to Table 1.

The correspondence between the two designs associates a block of size 7 with a block of size 8 if and only if the difference of the corresponding codewords is in the class (i); i.e., the support of the difference is a 3-flat.

We recall a few facts from the affine geometry: every set of three points defines a unique 2-flat. Furthermore, any 2-flat and a fifth point determine a unique 3-flat. By (vi) in Lemma 3.3 a support of size 7 does not contain a 2-flat, and thus any four points within a block of size 7 can be uniquely completed to a 3-flat. The intersection of two 3-flats can have only 0, 1, 2, 4, or 8 points.

In Table 3 we describe all combinations that need to be considered. Everything else comes with the automorphisms as in section 6. Every combination has three rows: original codeword in the first row, the linking codeword with a 3-flat support in the second row, and their sum in the third row. By suitable positioning of 1's and 3's within a 3-flat we get the required connections. However, this is not a 1-1 correspondence, as we can sometimes associate a block of one design with several blocks of the other design. This number is indicated in the Comb column.

For example, in the first case in (1a) we can choose the position with 1 in the 3-flat from the two positions with 3's in the original codeword. One 2-flat within the 3-flat is indicated by underlining its coordinates. As the 3-flat and the sum intersect in 5 points the result has rank 1, and one of the 2-flats is underlined.

The frequencies in the right side can be taken from Table 1, and then the frequencies in the left side can be counted from the relations in the table. For example, in case (0a) we can construct one block of size 8 from one block of size 7. On the other hand, from one block of size 8 we can construct four blocks of size 7. Hence in this case there must be four times as many blocks in Theorem 4.5 as in Theorem 5.1.

TABLE 3
Structural dependence of blocks in Theorems 4.5 and 5.1

Case	Theorem 4.5 \rightarrow Theorem 5.1			Theorem 5.1 \rightarrow Theorem 4.5				
	Fix		Comb	Freq	Fix		Comb	Freq
(0a)	$\overline{111}$	1112 33313111 33111		$\frac{2(n-8)}{3}$	$\overline{111}$	31113 13333111 2111		$\frac{n-8}{6}$
	$\overline{111}$	1332 31133113 13113			$\overline{111}$	31113 13311133 2 133	3	
(1a)	$\overline{112}$	1133 3 331 1311 111 31311	2	$\frac{n-8}{3}$	$\overline{111}$	13113 1 3 331311 112 3 311		$\frac{2(n-8)}{3}$
	$\overline{112}$	1133 3 3113311 111 1 3311	2		$\overline{111}$	13113 1 1331133 112 1 133		
(1b)	$\overline{112}$	1111 1 333 3111 113 13111	4		$\overline{113}$	13111 3 1333111 112 1 111		
	$\overline{112}$	3333 1 111 1111 113 31111	4		$\overline{113}$	13111 3 3 333333 112 3 333		
(2a)	132	1113 3 3311311 131 1 1311	3	$\frac{n-8}{3}$	131	11113 1 3 331311 132 1 311	2	$\frac{n-8}{2}$
(2b)	312	1113 3 3311311 311 1 1311	3		311	11113 1 3 331311 312 1 311	2	
(3a)	132	3331 1 111 1111 133 11111			133	11111 3 3 333333 132 1 333	2	$\frac{n-8}{6}$
(3b)	312	3331 1 111 1111 313 11111			313	11111 3 3 333333 312 1 333	2	

9. Conclusions and further research. We have constructed several new infinite families of simple 3-designs from the codewords of Hamming weight 8 in the \mathbf{Z}_4 -Goethals codes. This was done by analyzing the low-weight codewords and counting the number of solutions to certain systems of equations over finite fields. In addition, we described a relation between the designs in Theorems 4.5 and 5.1.

This paper raises many questions and we state a few of them. Can one prove Conjecture 5.9? Can one generalize the results in this paper to the \mathbf{Z}_4 -Goethals-like codes? Can one find simple designs from codewords of a larger Hamming weight? Last but not least, What is the most general adaptation of the Assmus–Mattson theorem in the \mathbf{Z}_4 -domain? This work suggests that the lifting rank may play a key role.

Acknowledgments. This paper and the proof of the main theorem were inspired by the manuscript [18]. Thanks are also due to J. Lahtonen for several useful discussions and to an anonymous referee for valuable comments.

REFERENCES

[1] E. F. ASSMUS, JR., AND H. F. MATTSON, JR., *New 5-designs*, J. Combin. Theory, 6 (1969), pp. 122–151.

- [2] L. A. BASSALYGO AND V. A. ZINOVIEV, *A remark on uniformly packed codes*, Problems Inform. Transmission, 13 (1977), pp. 178–180.
- [3] E. R. BERLEKAMP, H. RUMSEY, AND G. SOLOMON, *On the solution of algebraic equations over finite fields*, Information and Control, 10 (1967), pp. 553–564.
- [4] A. BONNECAZE, E. RAINS, AND P. SOLÉ, *3-Colored 5-designs and \mathbf{Z}_4 -codes*, J. Statist. Plann. Inference, 86 (2000), pp. 349–368.
- [5] I. DUURSMAN, T. HELLESETH, C. RONG, AND K. YANG, *Split weight enumerators for the Preparata codes with applications to designs*, Des. Codes Cryptogr., 18 (1999), pp. 103–124.
- [6] T. A. GULLIVER AND M. HARADA, *Extremal double circulant Type II codes over \mathbf{Z}_4 and construction of 5-(24, 10, 36) designs*, Discrete Math., 194 (1999), pp. 129–137.
- [7] A. R. HAMMONS, JR., P. V. KUMAR, A. R. CALDERBANK, N. J. A. SLOANE, AND P. SOLÉ, *The \mathbf{Z}_4 -linearity of Kerdock, Preparata, Goethals, and related codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 301–319.
- [8] M. HARADA, *New 5-designs constructed from the lifted Golay code over \mathbf{Z}_4* , J. Combin. Des., 6 (1998), pp. 225–229.
- [9] T. HELLESETH AND P. V. KUMAR, *The algebraic decoding of the \mathbf{Z}_4 -linear Goethals code*, IEEE Trans. Inform. Theory, 41 (1995), pp. 2040–2048.
- [10] T. HELLESETH, P. V. KUMAR, AND K. YANG, *An infinite family of 3-designs from Preparata codes over \mathbf{Z}_4* , Des. Codes Cryptogr., 15 (1998), pp. 175–181.
- [11] T. HELLESETH, C. RONG, AND K. YANG, *New infinite families of 3-designs from Preparata codes over \mathbf{Z}_4* , Discrete Math., 195 (1999), pp. 139–156.
- [12] T. HELLESETH, C. RONG, AND K. YANG, *On t -designs from codes over \mathbf{Z}_4* , Discrete Math., 238 (2001), pp. 67–80.
- [13] T. HELLESETH AND V. ZINOVIEV, *On \mathbf{Z}_4 -linear Goethals codes and Kloosterman sums*, Des. Codes Cryptogr., 17 (1999), pp. 269–288.
- [14] T. HELLESETH AND V. ZINOVIEV, *On coset weight distributions of the \mathbf{Z}_4 -linear Goethals codes*, IEEE Trans. Inform. Theory, 47 (2001), pp. 1758–1772.
- [15] D. L. KREHER, *t -Designs, $t \geq 3$* , in The CRC Handbook of Combinatorial Designs, C. J. Colbourn and J. H. Dinitz, eds., CRC Press, Boca Raton, FL, 1996, pp. 47–66.
- [16] F. J. MACWILLIAMS AND N. J. A. SLOANE, *The Theory of Error-Correcting Codes*, 9th ed., North-Holland Math. Library 16, North-Holland, Amsterdam, 1996.
- [17] N. V. SEMAKOV, V. A. ZINOVIEV, AND G. V. ZAITSEV, *Uniformly close-packed codes*, Problems Inform. Transmission, 7 (1971), pp. 30–39.
- [18] D.-J. SHIN, P. V. KUMAR, AND T. HELLESETH, *3-Designs from the \mathbf{Z}_4 -Goethals codes via a new Kloosterman sum identity*, Des. Codes Cryptogr., submitted.
- [19] D.-J. SHIN, P. V. KUMAR, AND T. HELLESETH, *An Assmus–Mattson-type approach for identifying 3-designs from linear codes over \mathbf{Z}_4* , Des. Codes Cryptogr., submitted.
- [20] D.-J. SHIN, P. V. KUMAR, AND T. HELLESETH, *5-Designs from the lifted Golay code over \mathbf{Z}_4 via an Assmus–Mattson type approach*, Discrete Math., 241 (2001), pp. 479–487.
- [21] K. TANABE, *An Assmus–Mattson theorem for \mathbf{Z}_4 -codes*, IEEE Trans. Inform. Theory, 46 (2000), pp. 48–53.
- [22] K. YANG AND T. HELLESETH, *Two new infinite families of 3-designs from Kerdock codes over \mathbf{Z}_4* , Des. Codes Cryptogr., 15 (1998), pp. 201–214.

OPTIMAL CONSECUTIVE- k -OUT-OF- n : G CYCLE FOR $n \leq 2k + 1$ *

DING-ZHU DU[†], FRANK K. HWANG[‡], XIAOHUA JIA[§], AND HUNG Q. NGO[¶]

Abstract. A cyclic consecutive- k -out-of- n : G system consists of n components lying on a cycle. Those components are exchangeable but may have different working probabilities. The system works if and only if there are k consecutive components at work. What is the optimal assignment of components for maximizing the reliability of the system? Does the optimal assignment depend on the working probability values of components? For $k \leq n \leq 2k + 1$, Zuo and Kuo in 1990 proposed a solution independent from the working probability values of components, called the invariant optimal assignment. However, their proof is incomplete, pointed out recently by Jalali et al. [*The Optimal Consecutive- k -out-of- n : G Line for $n \leq 2k$* , manuscript, 1999]. We present a complete proof in this paper.

Key words. invariant optimal assignment, consecutive- k -out-of- n : G cycle

AMS subject classifications. 60K10, 90B25

PII. S0895480100375041

1. Introduction. A cyclic consecutive- k -out-of- n : G system $con_C(k, n : G)$ is a cycle of $n (\geq k)$ components such that the system works if and only if some k consecutive components all work. Suppose n components with working probabilities $p_{[1]} \leq p_{[2]} \leq \dots \leq p_{[n]}$ are all exchangeable. How can they be assigned to the n positions on the cycle to maximize the reliability of the system? Kuo, Zhang, and Zuo [10] showed that if $k = 2$, then the optimal assignment is *invariant*; i.e., it depends only on the ordering of working probabilities of the components but not their value. They also claimed that for $k \geq 3$ and $n > 2k + 1$, $con_C(k, n : G)$ has no invariant optimal assignment. For $n \leq 2k + 1$, Zuo and Kuo [13] claimed that there exists an invariant optimal assignment

$$(p_{[1]}, p_{[3]}, p_{[5]}, \dots, p_{[6]}, p_{[4]}, p_{[2]}).$$

However, Jalali et al. [9] found that their proof is incomplete. A proof in case $n = 2k + 1$ was given in [6]. In this paper, we give a complete proof for this invariant optimal assignment with $n \leq 2k + 1$.

2. Main result. In this section, we show the following.

THEOREM 2.1. *For $k \leq n \leq 2k + 1$, there exists an invariant optimal assignment*

$$(p_{[1]}, p_{[3]}, p_{[5]}, \dots, p_{[6]}, p_{[4]}, p_{[2]}).$$

*Received by the editors July 11, 2000; accepted for publication (in revised form) March 18, 2002; published electronically May 8, 2002.

<http://www.siam.org/journals/sidma/15-3/37504.html>

[†]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (dzd@cs.umn.edu). This author's research was supported in part by the National Science Foundation under grant CCR-9530306, by the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, People's Republic of China, and in part by National 973 Information Technology and High-Performance Software Program of China under grant G1998030402.

[‡]Department of Applied Mathematics, National Chiao-Tung University, Hsinchu, Taiwan 9 (fhwang@math.nctu.edu.tw)

[§]Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong (jia@cs.cityu.edu.hk).

[¶]Department of Computer Science, State University of New York, Buffalo, NY 14260 (hungngo@cse.buffalo.edu).

Let p_1, p_2, \dots, p_n be reliabilities of the n components on the cycle in the counter-clockwise direction. For simplicity of the proof, we first assume that

$$0 < p_{[1]} < p_{[2]} < \dots < p_{[n]} < 1.$$

Our proof is based on the following representation of the reliability of consecutive- k -out-of- n : G cycle for $n \leq 2k + 1$.

LEMMA 2.2. *The reliability of consecutive- k -out-of- n : G cycle for $n \leq 2k + 1$ under assignment C can be represented as*

$$\begin{aligned} R(C) &= p_1 \cdots p_n + \sum_{i=1}^n q_i p_{i+1} \cdots p_{i+k} \\ &= p_1 \cdots p_n + \sum_{i=1}^n p_i \cdots p_{i+k-1} - \sum_{i=1}^n p_i \cdots p_{i+k}, \end{aligned}$$

where $q_i = 1 - p_i$ and $p_{n+i} = p_i$.

Proof. The system works if and only if all components work or for some i , the i th component fails, and the $(i + 1)$ st component, \dots , the $(i + k)$ th component all work. Since $n \leq 2k + 1$, there exists at most one such i . Therefore,

$$R(C) = p_1 \cdots p_n + \sum_{i=1}^n q_i p_{i+1} \cdots p_{i+k}.$$

Note that

$$q_i p_{i+1} \cdots p_{i+k} = p_{i+1} \cdots p_{i+k} - p_i \cdots p_{i+k}.$$

This implies the second representation. \square

This representation is a key point to show the main theorem. It explains why invariant optimal assignment exists for $n \leq 2k + 1$ but does not exist for $n > 2k + 1$.

For $n = k, k + 1$, by Lemma 2.2, $R(C)$ has the same value for all assignment C , and hence Theorem 2.1 is trivially true. Next, we assume $k + 2 \leq n \leq 2k + 1$.

To prove Theorem 2.1, it suffices to show that in any optimal assignment,

$$(2.1) \quad (p_i - p_j)(p_{i-1} - p_{j+1}) > 0 \quad \text{for } 1 < i < j < n.$$

In fact, the optimal assignment described in Theorem 2.1 can be determined uniquely by condition (2.1) (see [9]). Selecting any component to be labeled p_1 , we always have from condition (2.1) that

$$(2.2) \quad (p_i - p_{n-i+1})(p_{i+1} - p_{n-i}) > 0 \quad \text{for } i = 1, \dots, h,$$

where $h = \lfloor n/2 \rfloor$. For simplicity of representation, we denote $i' = n - i + 1$. When n is odd, $(\frac{n+1}{2})' = \frac{n+1}{2}$. Furthermore, without loss of generality, we assume $p_1 > p_{1'}$ throughout this proof. Then the condition (2.2) can be rewritten as

$$p_i > p_{i'} \quad \text{for } i = 1, \dots, h.$$

Let $I = \{i \mid 1 < i \leq h, p_i < p_{i'}\}$. Let C_I be the assignment obtained from C by exchanging components i and i' for all $i \in I$. To prove (2.1), it suffices to show that for any assignment C , if $I \neq \emptyset$,

$$R(C) < R(C_I).$$

Denote $I' = \{i' \mid i \in I\}$ and

$$(y_{i_1} \cdots y_{i_d})^I = \left(\prod_{1 \leq j \leq d, i_j \notin I \cup I'} y_{i_j} - \prod_{1 \leq j \leq d, i_j \in I \cup I'} y_{i'_j} \right) \left(\prod_{1 \leq j \leq d, i_j \in I \cup I'} y_{i'_j} - \prod_{1 \leq j \leq d, i_j \notin I \cup I'} y_{i_j} \right),$$

where $y_i = p_i$ or q_i . It is easy to verify that

$$\begin{aligned} & \left(y_{i_1} \cdots y_{i_d} \prod_{1 \leq j \leq d, i_j \in I \cup I'} \frac{y_{i'_j}}{y_{i_j}} + y_{i'_1} \cdots y_{i'_d} \prod_{1 \leq j \leq d, i_j \notin I \cup I'} \frac{y_{i_j}}{y_{i'_j}} \right) \\ & - (y_{i_1} \cdots y_{i_d} + y_{i'_1} \cdots y_{i'_d}) \\ & = (y_{i_1} \cdots y_{i_d})^I. \end{aligned}$$

Denote $Q_k(C) = \sum_{i=1}^n p_i \cdots p_{i+k-1}$. Then

$$R(C) = p_1 \cdots p_n + Q_k(C) - Q_{k+1}(C).$$

Let $a = \lfloor k/2 \rfloor$ and

$$s = \begin{cases} h & \text{if } n \text{ is even and } k \text{ is odd,} \\ h + 1 & \text{otherwise.} \end{cases}$$

Then we have the following.

LEMMA 2.3. $Q_k(C_I) - Q_k(C) = \sum_{i=1}^s (p_{-a+i} \cdots p_{-a+i+k-1})^I$.

Proof. Consider four cases.

Case 1. n and k both are even. In this case, $s = h + 1 = 1 + n/2$ and $a = k/2$.

Note that

$$\begin{aligned} & \sum_{i=h+2}^n p_{-a+i} \cdots p_{-a+i+k-1} \\ & = \sum_{i=2}^h p_{-a+(2h+2-i)} \cdots p_{-a+(2h+2-i)+k-1} \\ & = \sum_{i=2}^h p_{n-(-a+i+k-1)+1} \cdots p_{n-(-a+i)+1} \\ & = \sum_{i=2}^h p_{(-a+i+k-1)'} \cdots p_{(-a+i)'}. \end{aligned}$$

Thus,

$$\begin{aligned} & Q_k(C) \\ & = p_{-a+1} \cdots p_a + \sum_{i=2}^h (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)'}) \\ & \quad + p_{-a+h+1} \cdots p_{-a+h+k} \\ & = \prod_{j=1}^a (p_j p_{j'}) + \sum_{i=2}^h (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)'}) \\ & \quad + \prod_{j=1}^a (p_{h-a+j} p_{(h-a+j)'}). \end{aligned}$$

So,

$$Q_k(C_I) - Q_k(C) = \sum_{i=2}^h (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

However,

$$(p_{-a+1} \cdots p_{-a+k})^I = (p_{-a+h+1} \cdots p_{-a+h+k})^I = 0.$$

Therefore,

$$Q_k(C_I) - Q_k(C) = \sum_{i=1}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

Case 2. n is even and k is odd. In this case, $s = h = n/2$ and $a = (k-1)/2$. Note that

$$Q_k(C) = \sum_{i=1}^h (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)}').$$

Therefore,

$$Q_k(C_I) - Q_k(C) = \sum_{i=1}^h (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

Case 3. n and k both are odd. In this case, $s = h+1 = (n+1)/2$ and $a = (k-1)/2$. Note that

$$\begin{aligned} & Q_k(C) \\ &= \sum_{i=1}^h (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)}') + p_{-a+h+1} \cdots p_{-a+h+k} \\ &= \sum_{i=1}^h (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)}') \\ &\quad + p_{-a+h-1} p_{(-a+h-1)'} \cdots p_h p_{h'} p_{h+1}. \end{aligned}$$

Thus,

$$Q_k(C_I) - Q_k(C) = \sum_{i=1}^h (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

However,

$$(p_{-a+h+1} \cdots p_{-a+h+k})^I = 0.$$

Therefore,

$$Q_k(C_I) - Q_k(C) = \sum_{i=1}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

Case 4. n is odd and k is even. In this case, $s = h + 1 = (n + 1)/2$ and $a = k/2$. Note that

$$\begin{aligned} Q_k(C) &= p_{-a+1} \cdots p_{-a+k} + \sum_{i=2}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)'}) \\ &= \prod_{j=1}^a (p_j p_{j'}) + \sum_{i=2}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1} + p_{(-a+i)'} \cdots p_{(-a+i+k-1)'}). \end{aligned}$$

Thus

$$Q_k(C_I) - Q_k(C) = \sum_{i=2}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1})^I.$$

However,

$$(p_{-a+1} \cdots p_{-a+k})^I = 0.$$

Therefore,

$$Q_k(C_I) - Q_k(C) = \sum_{i=1}^{h+1} (p_{-a+i} \cdots p_{-a+i+k-1})^I. \quad \square$$

Define

$$t = \begin{cases} h & \text{if } n \text{ is even and } k + 1 \text{ is odd,} \\ h + 1 & \text{otherwise} \end{cases}$$

and $b = \lfloor (k + 1)/2 \rfloor$. We have a useful representation of $R(C_I) - R(C)$ as follows.

LEMMA 2.4.

$$R(C_I) - R(C) = \sum_{i=2}^t (p_{-b+i} \cdots p_{-b+i+k-1})^I - \sum_{i=1}^t (p_{-b+i} \cdots p_{-b+i+k})^I.$$

Proof. By Lemma 2.3, we have

$$R(C_I) - R(C) = \sum_{i=1}^s (p_{-a+i} \cdots p_{-a+i+k-1})^I - \sum_{i=1}^t (p_{-b+i} \cdots p_{-b+i+k})^I.$$

Note that if k is even and n is odd, then $a = b$, $s = t$, and

$$(p_{-a+1} \cdots p_{-a+k})^I = (p_{-k/2+1} \cdots p_{k/2})^I = 0;$$

if k is odd and n is odd, then $a = b - 1$, $s = t$, and

$$(p_{-a+s} \cdots p_{-a+s+k-1})^I = (p_{-(n-k)/2} \cdots p_{-(n+k)/2-1})^I = 0;$$

if k is even and n is even, then $a = b$, $s = t + 1$, and

$$(p_{-a+1} \cdots p_{-a+k})^I = (p_{-a+s} \cdots p_{-a+s+k-1})^I = 0;$$

if k is odd and n is even, then $a = b - 1$ and $s = t - 1$. Therefore, we always have

$$\sum_{i=1}^s (p_{-a+i} \cdots p_{-a+i+k-1})^I = \sum_{i=2}^t (p_{-a+i} \cdots p_{-a+i+k-1})^I. \quad \square$$

Note that $(p_{-b+i} \cdots p_{-b+i+k-1})^I \geq 0$ for $2 \leq i \leq t$ and $(p_{-b+i} \cdots p_{-b+i+k})^I \geq 0$ for $1 \leq i \leq t$. Therefore, to prove $R(C_I) < R(C)$, we need to compare $(p_{-b+i} \cdots p_{-b+i+k-1})^I$ with $(p_{-b+i} \cdots p_{-b+i+k})^I$.

LEMMA 2.5. *Suppose $I = \{i \mid 1 < i \leq h, p_i < p_{i'}\}$. Then, for $i = 1, \dots, b$,*

$$(q_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I = (p_{-b+i+1} \cdots p_{-b+i+k})^I - (p_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I \geq 0,$$

and the strict inequality sign holds if and only if

$$\begin{aligned} \{j \mid b - i + 1 \leq j \leq \min(-b + i + k, n + b - i - k), j \in I\} &\neq \emptyset, \\ \{j \mid b - i + 1 \leq j \leq \min(-b + i + k, n + b - i - k), j \notin I\} &\neq \emptyset. \end{aligned}$$

Proof. First, assume $-b + i \in I \cup I'$. Then we have

$$\begin{aligned} &(q_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I \\ &= \left(q_{-b+i} \prod_{-b+i+1 \leq j \leq -b+i+k, j \notin I \cup I'} p_j - q_{(-b+i)'} \prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_{j'} \right) \\ &\quad \cdot \left(\prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_{j'} - \prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_j \right) \\ &= \left(\prod_{-b+i+1 \leq j \leq -b+i+k, j \notin I \cup I'} p_j - \prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_{j'} \right) \\ &\quad \cdot \left(\prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_{j'} - \prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_j \right) \\ &\quad - \left(\prod_{-b+i \leq j \leq -b+i+k, j \notin I \cup I'} p_j - \prod_{-b+i \leq j \leq -b+i+k, j \notin I \cup I'} p_{j'} \right) \\ &\quad \cdot \left(\prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_{j'} - \prod_{-b+i+1 \leq j \leq -b+i+k, j \in I \cup I'} p_j \right) \\ &= (p_{-b+i+1} \cdots p_{-b+i+k})^I - (p_i \cdots p_{i+k})^I. \end{aligned}$$

If $-b + i + k < n + b - i - k + 1$, then $-b + i + k \leq h$, and hence we have

$$\begin{aligned} &(q_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I \\ &= \left(\prod_{j=1}^{b-i} p_j p_{j'} \right) \left(\prod_{b-i+2 \leq j \leq -b+i+k, j \notin I} p_j - \prod_{b-i+1 \leq j \leq -b+i+k, j \notin I} p_{j'} \right) \\ &\quad \left(p_{b-i+1} q_{(b-i+1)'} \prod_{b-i+1 \leq j \leq -b+i+k, j \in I} p_{j'} - p_{(b-i+1)'} q_{b-i+1} \prod_{b-i+1 \leq j \leq -b+i+k, j \in I} p_j \right) \\ &\geq 0 \end{aligned}$$

since $-b + i \in I \cup I'$ implies that

$$p_{b-i+1}q_{(b-i+1)'} - p_{(b-i+1)'}q_{b-i+1} = p_{b-i+1} - p_{(b-i+1)'} > 0.$$

Moreover, it is easy to verify that

$$(q_{-b+i}p_{-b+i+1} \cdots p_{-b+i+k})^I > 0$$

if and only if

$$\begin{aligned} \{j \mid b-i \leq j \leq -b+i+k, j \in I\} &\neq \emptyset, \\ \{j \mid b-i \leq j \leq -b+i+k, j \notin I\} &\neq \emptyset. \end{aligned}$$

If $-b+i+k \geq n+b-i-k+1$, then $n+b-i-k \leq h$, and hence

$$\begin{aligned} &(q_{-b+i}p_{-b+i+1} \cdots p_{-b+i+k})^I \\ &= \left(\prod_{j=1}^{b-i} p_j p_{j'} \right) \left(\prod_{b-i+2 \leq j \leq n+b-i-k, j \notin I} p_j - \prod_{b-i+1 \leq j \leq n+b-i-k, j \in I} p_{j'} \right) \\ &\quad \left(p_{b-i+1}q_{(b-i+1)'} \prod_{b-i+1 \leq j \leq n+b-i-k, j \in I} p_{j'} - p_{(b-i+1)'}q_{b-i+1} \prod_{b-i+1 \leq j \leq n+b-i-k, j \notin I} p_j \right) \\ &\quad \cdot \left(\prod_{n+b-i-k+1 \leq j \leq h} p_j p_{j'} \right) \gamma \\ &\geq 0, \end{aligned}$$

where

$$\gamma = \begin{cases} 1 & \text{if } n \text{ is even,} \\ p_{h+1} & \text{if } n \text{ is odd.} \end{cases}$$

Moreover,

$$(q_{-b+i}p_{-b+i+1} \cdots p_{-b+i+k})^I > 0$$

if and only if

$$\begin{aligned} \{j \mid b-i+1 \leq j \leq n+b-i-k, j \in I\} &\neq \emptyset, \\ \{j \mid b-i+1 \leq j \leq n+b-i-k, j \notin I\} &\neq \emptyset. \end{aligned}$$

Finally, we note that a similar argument works in the case that $-b+i \notin I \cup I'$. \square

Similarly, we can show the following.

LEMMA 2.6. *Suppose $I = \{i \mid 1 \leq i \leq h, p_i < p_i'\}$. Then, for $i = b, \dots, t$,*

$$\begin{aligned} (p_{-b+i} \cdots p_{-b+i+k-1} q_{-b+i+k})^I &= (p_{-b+i} \cdots p_{-b+i+k-1})^I - (p_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I \\ &\geq 0. \end{aligned}$$

LEMMA 2.7. *Suppose $I = \{i \mid 1 \leq i \leq h, p_i < p_{i'}\}$. Then*

$$\begin{aligned} & (q_0 p_1 \cdots p_k q_{k+1})^I \\ &= (p_1 \cdots p_k)^I - (p_0 \cdots p_k)^I - (p_1 \cdots p_{k+1})^I + (p_0 \cdots p_{k+1})^I \\ &\geq 0, \end{aligned}$$

and the strict inequality sign holds if and only if

$$\begin{aligned} & \{j \mid 1 \leq j \leq n-k, j \in I\} \neq \emptyset, \\ & \{j \mid 1 \leq j \leq n-k, j \notin I\} \neq \emptyset. \end{aligned}$$

By Lemmas 2.4–2.7, we have

$$\begin{aligned} & R(C_I) - R(C) \\ &= \sum_{i=1}^{b-1} (q_{-b+i} p_{-b+i+1} \cdots p_{-b+i+k})^I + \sum_{i=b+2}^t (p_{-b+i} \cdots p_{-b+i+k-1} q_{-b+i+k})^I \\ &+ (q_0 p_1 \cdots p_k q_{k+1})^I - \left(\prod_{j=0}^{k+1} p_j \right)^I. \end{aligned}$$

Let $d = \lceil (n-k)/2 \rceil - 1$. Note that

$$\begin{aligned} & \left(\prod_{j=0}^{k+1} p_j \right)^I \\ &= \left(q_{-1} \prod_{j=0}^{k+1} p_j \right)^I + \left(\prod_{j=-1}^{k+1} p_j \right)^I \\ &= \cdots \\ &= \sum_{i=1}^d \left(q_{-i} \prod_{j=-i+1}^{k+i} p_j \right)^I + \sum_{i=1}^{n-k-2-d} \left(q_{i+k+1} \prod_{j=-i}^{i+k} p_j \right)^I + \left(\prod_{j=-d}^{n-d-1} p_j \right)^I \end{aligned}$$

and

$$\left(\prod_{j=-d}^{n-d-1} p_j \right)^I = 0.$$

Thus, we have

$$\begin{aligned} (2.3) \quad & R(C_I) - R(C) \\ &= \sum_{i=1}^{b-d-1} \left(q_{-b+i} \prod_{j=-b+i+1}^{-b+i+k} p_j \right)^I \\ &+ \sum_{i=1}^d \left[\left(q_{-i} \prod_{j=-i+1}^{-i+k} p_j \right)^I - \left(q_{-i} \prod_{j=-i+1}^{i+k} p_j \right)^I \right] \end{aligned}$$

$$\begin{aligned}
& + \sum_{i=b+n-k-d}^t \left(q_{-b+i+k} \prod_{j=-b+i}^{-b+i+k-1} p_j \right)^I \\
& + \sum_{i=1}^{n-k-2-d} \left[\left(q_{i+k+1} \prod_{j=i+1}^{i+k} p_j \right)^I - \left(q_{i+k+1} \prod_{j=-i}^{i+k} p_j \right)^I \right] \\
& + (q_0 p_1 \cdots p_k q_{k+1})^I.
\end{aligned}$$

This representation suggests that we show Lemmas 2.8 and 2.9.

LEMMA 2.8. For $i = 1, \dots, d$,

$$\left(q_{-i} \prod_{j=-i+1}^{-i+k} p_j \right)^I \geq \left(q_{-i} \prod_{j=-i+1}^{i+k} p_j \right)^I,$$

and the inequality holds strictly if and only if

$$\left(q_{-i} \prod_{j=-i+1}^{-i+k} p_j \right)^I > 0.$$

Proof. First, consider the case that $-i \in I'$ and n is even. Denote

$$\begin{aligned}
A &= \prod_{-i+1 \leq j \leq -i+k, j \notin I \cup I'} p_j - \prod_{-i+1 \leq j \leq -i+k, j \notin I \cup I'} p_{j'}, \\
B &= \prod_{-i+1 \leq j \leq i+k, j \notin I \cup I'} p_j - \prod_{-i+1 \leq j \leq i+k, j \notin I \cup I'} p_{j'}, \\
A' &= q_{(-i)'} \prod_{-i+1 \leq j \leq -i+k, j \in I \cup I'} p_{j'} - q_{-i} \prod_{-i+1 \leq j \leq -i+k, j \notin I \cup I'} p_j, \\
B' &= q_{(-i)'} \prod_{-i+1 \leq j \leq i+k, j \in I \cup I'} p_{j'} - q_{-i} \prod_{-i+1 \leq j \leq -i+k, j \notin I \cup I'} p_j.
\end{aligned}$$

Then

$$\left(q_{-i} \prod_{j=-i+1}^{-i+k} p_j \right)^I - \left(q_{-i} \prod_{j=-i+1}^{i+k} p_j \right)^I = AA' - BB'.$$

If $-i+k < n+i-k+1$, then $-i+k \leq h$. Hence

$$\begin{aligned}
& A - B \\
&= \left(\prod_{1 \leq j \leq i, j \notin I} p_j p_{j'} \right) \left[\left(\prod_{i+1 \leq j \leq n-i-k, j \notin I} p_j \right) \alpha(1 - \beta\delta) \right. \\
&\quad \left. - \left(\prod_{i+1 \leq j \leq n-i-k, j \notin I} p_{j'} \right) \beta(1 - \alpha\delta) \right],
\end{aligned}$$

where

$$\begin{aligned}\alpha &= \prod_{n-i-k+1 \leq j \leq -i+k, j \notin I} p_j, \\ \beta &= \prod_{n-i-k+1 \leq j \leq -i+k, j \notin I} p_{j'}, \\ \delta &= \prod_{-i+k+1 \leq j \leq h, j \notin I} p_j p_{j'},\end{aligned}$$

and

$$\alpha(1 - \beta\delta) - \beta(1 - \alpha\delta) = \alpha - \beta \geq 0.$$

Thus, $A \geq B$.

If $-i + k \geq n + i - k + 1$, then $n + i - k \leq h$. Hence

$$\begin{aligned} & A - B \\ &= \left(\prod_{1 \leq j \leq i, j \notin I} p_j p_{j'} \right) \left(\prod_{n+i-k+1 \leq j \leq h, j \notin I} p_j p_{j'} \right) \left[\left(\prod_{i+1 \leq j \leq n-i-k, j \notin I} p_j \right) \alpha(1 - \beta) \right. \\ & \quad \left. - \left(\prod_{i+1 \leq j \leq n-i-k, j \notin I} p_{j'} \right) \beta(1 - \alpha) \right], \end{aligned}$$

where

$$\begin{aligned}\alpha &= \prod_{n-i-k+1 \leq j \leq n+i-k, j \notin I} p_j, \\ \beta &= \prod_{n-i-k+1 \leq j \leq n+i-k, j \notin I} p_{j'}\end{aligned}$$

and

$$\alpha(1 - \beta) - \beta(1 - \alpha) = \alpha - \beta \geq 0.$$

Thus, $A \geq B$.

Similarly, we have $A' \geq B'$. Therefore, $AA' \geq BB'$. By similar arguments, we can prove the inequalities in other cases.

Now it is easy to verify that $AA' > BB'$ if and only if

$$\begin{aligned}\{j \mid i + 1 \leq j \leq \min(-i + k, n + i - k), h \in I\} &\neq \emptyset, \\ \{j \mid i + 1 \leq j \leq \min(-i + k, n + i - k), h \notin I\} &\neq \emptyset\end{aligned}$$

if and only if, or, by Lemma 2.5,

$$\left(q_{-i} \prod_{j=-i+1}^{-i+k} p_j \right)^I > 0. \quad \square$$

Similarly, we can prove the following.

LEMMA 2.9. For $i = 1, \dots, n - k - 2 - d$,

$$\left(q_{i+k+1} \prod_{j=i+1}^{i+k} p_j \right)^I \geq \left(q_{i+k+1} \prod_{j=-i}^{i+k} p_j \right)^I.$$

By Lemmas 2.5–2.9, all terms in the right-hand side of (2.3) are nonnegative. Next, we show that if $I \neq \emptyset$, then at least one term in (2.3) is positive.

Note that $1 \notin I$. Since $I \neq \emptyset$, there exists a positive integer r such that $1 \leq r < h$, $r \notin I$, and $r + 1 \in I$. If $r + 1 \leq n - k$, then

$$\begin{aligned} r &\in \{j \mid 1 \leq j \leq n - k, j \notin I\}, \\ r + 1 &\in \{j \mid 1 \leq j \leq n - k, j \in I\}, \end{aligned}$$

and hence

$$q_0 p_1 \cdots p_k q_{k+1} > 0.$$

If $r + 1 > n - k$, then choose $i = b + n - k - (r + 1) < b$, and we have

$$b - i + 1 \leq r, r + 1 \leq \min(-b + i + k, n + b - i - k).$$

Hence, by Lemma 2.5,

$$q_i p_{i+1} \cdots p_{i+k} > 0.$$

Finally, we deal with the case that some equality signs hold in $0 \leq p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]} \leq 1$. If $p_{[1]} = p_{[2]} = \cdots = p_{[n]}$, then Theorem 2.1 is trivially true. If $p_{[i]} < p_{[i+1]}$, then, for sufficiently small $\varepsilon > 0$, we have

$$0 < p_{[1]} + \varepsilon < \cdots < p_{[i]} + i\varepsilon < p_{[i+1]} - (n - i)\varepsilon < \cdots < p_{[n]} - \varepsilon < 1.$$

For them, we already proved the optimality of assignment C^* in Theorem 2.1; that is, for any assignment C , $R(C^*) \geq R(C)$. Now we can complete our proof of Theorem 2.1 by setting $\varepsilon \rightarrow 0$.

3. Discussion. An invariant optimal assignment is a nice thing to have in practice and also an interesting mathematical problem to solve. The existence of an invariant optimal assignment has been widely studied for the consecutive- k -out-of- n : F systems and G systems, where a F system works if and only if there do not exist k consecutive components that all fail. Usually, the nonexistence of invariant optimal assignments was demonstrated [12, 7, 13]. There are only four nontrivial cases that invariant optimal assignments may exist. The first is an invariant optimal assignment for the consecutive-2-out-of- n : F line conjectured by Derman, Lieberman, and Ross [2] and independently proved by Du and Hwang [3] and Malon [11]. In fact, the former proved the harder cycle version which is the second case of existence. Note that the cycle version implies the line version since by setting $p_{[n]} = 1$ ($p_{[1]} = 0$ in the G system), the line problem is reduced to the cycle problem. The third case is an invariant optimal assignment for the consecutive- k -out-of- n : G line for $n \leq 2k$ conjectured by Kuo, Zhang, and Zuo [10], and proved by Jalali et al. [9]. The fourth case is its cycle version, the current case. Note that again the cycle version implies the line version but is much harder. In the line version, one needs only to prove the case

$n = 2k$, and the $n < 2k$ case can be reduced to the $n = 2k$ case. No similar reduction is possible for the cycle case. One may wonder whether a simpler proof exists by considering other pairings. In the current paper, we break the term $q_i p_{i+1} \cdots p_{i+k}$ into two parts, $p_{i+1} \cdots p_{i+k}$ and $-p_i \cdots p_{i+k}$. Use the pairing of $p_{-a+i} \cdots p_{-a+i+k-1}$ with $p_{(-a+i)'} \cdots p_{(-a+i+k-1)'}$ for the first part and a similar one for the second part; then compare the C assignment with the C_I assignment. However, since the comparison of one part is positive and the other is negative, we have to further compare their sizes, thus complicating the proof. Can we not break the term $q_i p_{i+1} \cdots p_{i+k}$ and find a pairing to work? One such possibility is also to consider the clockwise representation of $R(C)$, namely, $R(C) = p_1 \cdots p_n + \sum_{i=1}^n q_i p_{i-1} \cdots p_{i-k}$. We then pair each term $q_i p_{i+1} \cdots p_{i+k}$ from the counterclockwise representation with the term $q_{i'} p_{(i+1)'} \cdots p_{(i+k)'}$ from the clockwise representation. The C_I assignment is better than the C assignment in all cases except when $q_i < q_{i'}$ and i' does not belong to $i, \dots, i+k$. The determination of invariant optimal assignments on lines and cycles is an application of the broader problem of finding an optimal permutation, linear or cyclic, under a certain objective function. This type of problem has been considered before [1, 8] when the arguments of the objective function are $|x_i - x_{i+1}|$ for all i . In the optimal assignment problem, the arguments are products like $q_i p_{i+1} \cdots p_{i+k}$, which seems to raise a new type of optimal permutation problem. In this paper we give a solution to one such problem and hope the approach may work for other similar problems [1, 8, 4, 5].

REFERENCES

- [1] C.-C. CHAO AND W.-Q. LIANG, *Arranging N distinct numbers on a line or a circle to reach extreme total variations*, European J. Combin., 13 (1992) pp. 325–334.
- [2] C. DERMAN, G.J. LIEBERMAN, AND S.M. ROSS, *On the consecutive-2-out-of- n : F system*, IEEE Trans. Rel., 31 (1982), pp. 57–63.
- [3] D.-Z. DU AND F.K. HWANG, *Optimal consecutive-2-out-of- n systems*, Math. Oper. Res., 11 (1986), pp. 187–191.
- [4] D.-Z. DU AND K.-I. KO, *Some completeness results on decision trees and group testing*, SIAM J. Alg. Discr. Meth., 8 (1987), pp. 762–777.
- [5] D.Z. DU AND F.K. HWANG, *Optimal assembly of an s -stage k -out-of- n system*, SIAM J. Discrete Math., 3 (1990), pp. 349–354.
- [6] D.-Z. DU, F.K. HWANG, Y. JUNG, AND H.Q. NGO, *Optimal consecutive- k -out-of- $(2k+1)$: G cycle*, J. Global Optim., 19 (2001), pp. 51–60.
- [7] F.K. HWANG, *Invariant permutations for consecutive- k -out-of- n cycles*, IEEE Trans. Rel., 38 (1989), pp. 65–67.
- [8] F.K. HWANG, *Extreme permutations with respect to weak majorizations*, European J. Combin., 17 (1996), pp. 637–645.
- [9] A. JALALI, A.G. HAWKES, L. CUI, AND F.K. HWANG, *The Optimal Consecutive- k -out-of- n : G Line for $n \leq 2k$* , manuscript, 1999.
- [10] W. KUO, W. ZHANG, AND M. ZUO, *A consecutive- k -out-of- n : G system: The mirror image of a consecutive- k -out-of- n : F system*, IEEE Trans. Rel., 39 (1990), pp. 244–253.
- [11] D.M. MALON, *Optimal consecutive-2-out-of- n : F component sequencing*, IEEE Trans. Rel., 33 (1984), pp. 414–418.
- [12] D.M. MALON, *Optimal consecutive- k -out-of- n : F component sequencing*, IEEE Trans. Rel., 34 (1985), pp. 46–49.
- [13] M. ZUO AND W. KUO, *Design and performance analysis of consecutive- k -out-of- n structure*, Naval Res. Logist., 37 (1990), pp. 203–230.

FAULT-TOLERANT EMBEDDINGS OF HAMILTONIAN CIRCUITS IN K -ARY N -CUBES*

YAAGOUB A. ASHIR[†] AND IAIN A. STEWART[†]

Abstract. We consider the fault-tolerant capabilities of networks of processors whose underlying topology is that of the k -ary n -cube Q_n^k , where $k \geq 3$ and $n \geq 2$. In particular, given a copy of Q_n^k where some of the interprocessor links may be faulty but where every processor is incident with at least two healthy links, we show that if the number of faults is at most $4n - 5$, then Q_n^k still contains a Hamiltonian circuit, but that there are situations where the number of faults is $4n - 4$ (and every processor is incident with at least two healthy links) and no Hamiltonian circuit exists. We also remark that given a faulty Q_n^k , the problem of deciding whether there exists a Hamiltonian circuit is NP-complete.

Key words. Hamiltonian circuits, embeddings, fault-tolerance, k -ary n -cubes, NP-completeness

AMS subject classifications. 68R10, 05C45

PII. S0895480196311183

1. Introduction. The hypercube or, more precisely, the binary n -cube B_n (where $n \geq 2$), is a popular interconnection network for parallel processing as it possesses a number of topological properties which are highly desirable in the context of parallel processing: for example, it contains a Hamiltonian circuit; many other networks can be embedded into a binary n -cube; and its symmetry results in rich communication properties (see, for example, [3, 5, 8, 10, 12] and the references therein).

Fault-tolerance in the binary n -cube is an important issue, given that many other networks can be embedded therein, and has been studied in a number of contexts. For example, the ability of the binary n -cube to route and reconfigure itself in spite of faults has been considered (see the references in [8]), as has the embedding of Hamiltonian circuits in binary n -cubes in the presence of faults [8]. In particular, Chan and Lee [8] proved that a binary n -cube where at most $2n - 5$ links are faulty and where every node is incident with at least two healthy links (a natural assumption to make) has a Hamiltonian circuit, but that there exist binary n -cubes with $2n - 4$ faults (and where every node is incident with at least two healthy links) not containing a Hamiltonian circuit. It is with an analogous version of this result that we are concerned in this paper.

One drawback of the binary n -cube is that the number of links incident with each node is logarithmic in the number of nodes, and this causes problems with regard to current VLSI technology when the networks built upon the binary n -cube topology involve a large number of processors. One means proposed to alleviate this problem is to base networks on the topology of the k -ary n -cube Q_n^k (where $k \geq 3$ and $n \geq 2$). A network based on Q_n^k is such that each node is incident with $2n$ links, and consequently k can be increased, in order to incorporate more processors, at the same time keeping n constant. Moreover, “high-dimensional” networks generally cost more

*Received by the editors October 28, 1996; accepted for publication (in revised form) March 22, 2002; published electronically May 8, 2002. The research of the first author was supported by the University of Bahrain.

<http://www.siam.org/journals/sidma/15-3/31118.html>

[†]Department of Mathematics and Computer Science, Leicester University, Leicester LE1 7RH, UK (y.ashir@mcs.le.ac.uk, i.a.stewart@mcs.le.ac.uk).

and run more slowly than “low-dimensional” networks, and it has also been shown that low-dimensional networks achieve lower latency and better hot-spot throughput than their high-dimensional counterparts [9, 11].

The properties of the k -ary n -cube Q_n^k relevant to parallel processing have not been determined to such an extent as those of the binary n -cube: however, some work has been done (see, for example, [1, 2, 4, 6, 7]). In particular, it has been shown that Q_n^k has a Hamiltonian circuit [6].

In this paper, we examine the number of link faults that a k -ary n -cube Q_n^k can tolerate so that there is still a Hamiltonian circuit. (Of course, we assume that every node is incident with at least two healthy links.) In particular, we show that a k -ary n -cube Q_n^k where at most $4n - 5$ links are faulty and where every node is incident with at least two healthy links has a Hamiltonian circuit, but that there exist k -ary n -cubes with $4n - 4$ faults (and where every node is incident with at least two healthy links) not containing a Hamiltonian circuit. We also remark that the general problem of deciding whether a faulty k -ary n -cube contains a Hamiltonian circuit is NP-complete for all (fixed) $k \geq 3$. Our results can be regarded as direct analogues of those in [8] for k -ary n -cubes as opposed to binary n -cubes.

2. Tolerating faults. Throughout this paper, we prefer to use the terminology “nodes” and “links” as opposed to “vertices” and “edges,” for whilst the results in this paper are entirely graph-theoretic, the use of “nodes” and “links” accentuates the motivational source of our research, i.e., the fault-tolerating capabilities of networks of processors when the faults which may occur are the failures of the links between processors in the network.

The binary n -cube, for $n \geq 2$, can be represented as the set of 2^n nodes $\{0, 1\}^n$ where there is a link joining nodes u and v if and only if u and v agree on all components except one. Note that each node has degree n . The k -ary n -cube Q_n^k , for $k \geq 2$ and $n \geq 2$, can be represented as the set of k^n nodes $\{0, 1, \dots, k - 1\}^n$ where there is a link joining nodes u and v if and only if u and v agree on all components except one, and on that component they differ by 1 modulo k . Note that each node has degree $2n$, when $k \geq 3$, and n when $k = 2$. In particular, Q_n^2 is simply B_n .

For each $i \in \{1, 2, \dots, n\}$, we refer to all links whose incident nodes differ in the i th component as lying in *dimension* i . Note that for any $i \in \{1, 2, \dots, n\}$, Q_n^k consists of k disjoint copies of Q_{n-1}^k where corresponding nodes are joined in circuits of length k using links in dimension i . When we consider Q_n^k in this way, with the disjoint copies joined by links lying in dimension i , we say that we have *partitioned* Q_n^k *over dimension* i .

Let us now proceed to the proof of our main theorem. This proof is by induction. We begin by proving the inductive step, and then we return to the base cases of the induction.

THEOREM 2.1. *Let $k \geq 4$ and $n \geq 2$, or let $k = 3$ and $n \geq 3$. If Q_n^k has at most $4n - 5$ faulty links and is such that every node is incident with at least 2 healthy links, then Q_n^k has a Hamiltonian circuit.*

Proof. The proof proceeds by induction on n . We handle the base cases, when $n = 2$ and $k \geq 4$ and when $n = 3$ and $k = 3$, later. As our induction hypothesis, assume that the result holds for Q_n^k , for some $n \geq 2$ and for all $k \geq 4$, or for some $n \geq 3$ and $k = 3$. Let Q_{n+1}^k have $4n - 1$ faults and be such that every node is incident with at least two healthy links. Then there exists some dimension, say dimension 1, which contains at least three faults. We can partition Q_{n+1}^k over dimension 1 and consider Q_{n+1}^k to consist of k disjoint copies Q_1, Q_2, \dots, Q_k of Q_n^k with corresponding

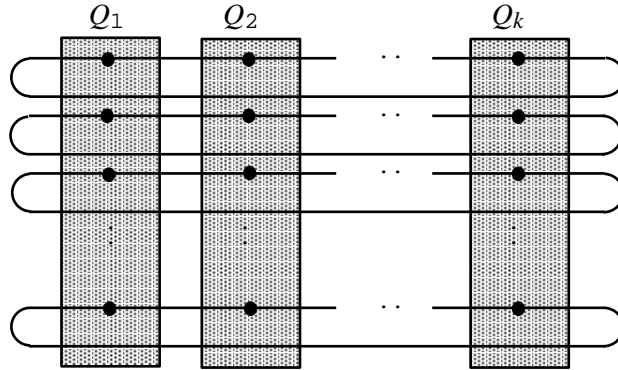


FIG. 2.1. The k copies of Q_n^k .

nodes joined in circuits of length k , where the faults contained in Q_1, Q_2, \dots, Q_k total at most $4n - 4$ (see Figure 2.1). Throughout this proof, if u is a node of Q_i , say, then we often denote it by u_i , and we refer to its corresponding node in Q_j as u_j . Our general aim below is to argue, using induction, that Hamiltonian circuits exist in each of Q_1, Q_2, \dots, Q_k and that we can “join” these circuits together using links in dimension 1 to obtain a Hamiltonian circuit in Q_{n+1}^k . (What we mean by “join” will become clear later: also, the general aim of connecting together circuits in Q_1, Q_2, \dots, Q_k actually has to be more sophisticated in some scenarios.) Naturally, different scenarios arise according to the distribution of faulty links in Q_1, Q_2, \dots, Q_k and in dimension 1. Another complication is that the chosen Hamiltonian circuit in Q_2 , for example, might depend upon the Hamiltonian circuit chosen in Q_1 .

Case (i). Each Q_i is such that every node is incident with at least two healthy links and no Q_i contains $4n - 4$ faults.

Without loss of generality (w.l.o.g.) we may assume that Q_1 has most faults from amongst Q_1, Q_2, \dots, Q_k . Hence, each of Q_2, Q_3, \dots, Q_k has at most $2n - 2$ faults. By the induction hypothesis, Q_1 has a Hamiltonian circuit C_1 . Following our basic strategy, outlined above, we wish to find a Hamiltonian circuit C_k in Q_k or a Hamiltonian circuit C_2 in Q_2 so that we might “join” such a Hamiltonian circuit to C_1 using healthy links in dimension 1. By “join” we mean replace a link (x_1, y_1) of C_1 and the (corresponding) link (x_2, y_2) of C_2 , for example, with the links (x_1, x_2) and (y_1, y_2) in dimension 1. However, we must ensure that two mutually compatible links exist in C_1 and C_2 and also that the relevant dimension 1 links are healthy.

We begin by applying a counting argument to show that there exist links (x_1, y_1) and (y_1, z_1) of C_1 such that either

- $(x_2, y_2), (y_2, z_2), (x_1, x_2), (y_1, y_2),$ and (z_1, z_2) are all healthy

or

- $(x_k, y_k), (y_k, z_k), (x_1, x_k), (y_1, y_k),$ and (z_1, z_k) are all healthy.

Suppose that it were otherwise. Then there would exist at least $2\lfloor k^n/3 \rfloor$ faults not in Q_1 . (Split C_1 into groups of three consecutive vertices and look at the links on either side in dimension 1 and in Q_2 and Q_k .) However, when $n \geq 2$ and $k \geq 4$ or when $n \geq 3$ and $k = 3$, we have that $2\lfloor k^n/3 \rfloor > 4n - 1$, which yields a contradiction. Hence, w.l.o.g. we may assume that there exist links (x_1, y_1) and (y_1, z_1) of C_1 such that $(x_2, y_2), (y_2, z_2), (x_1, x_2), (y_1, y_2),$ and (z_1, z_2) are all healthy.

What we need to do now is to show that there is a Hamiltonian circuit C_2 in Q_2 containing either (x_2, y_2) or (y_2, z_2) : we can then join C_1 and C_2 as described above. Suppose that it were otherwise. If necessary, mark some of the links of Q_2 incident with y_2 as faulty (that is, temporarily regard them as faulty) so that y_2 is incident with at most three healthy links in Q_2 , two of which are always (x_2, y_2) and (y_2, z_2) . Consequently, as there were originally at most $2n - 2$ faulty links in Q_2 , there are now at most $4n - 5$ faulty links. However, in order to apply our induction hypothesis (and deduce that this amended Q_2 has a Hamiltonian circuit), we need that every node in (the amended) Q_2 is incident with at least two healthy links. Suppose that it were otherwise. Then there is a node w_2 incident with exactly one healthy link. This must have been because (y_2, w_2) was a healthy link in the original Q_2 and it was subsequently marked as faulty. Amend the marking of healthy links so that (w_2, y_2) is the third healthy link in the amended Q_2 . Note that in the amended marking every node is incident with at least two healthy links (because Q_2 originally had at most $2n - 2$ faults). Now we can apply the induction hypothesis and deduce that Q_2 has a Hamiltonian circuit C_2 containing either (x_2, y_2) or (y_2, z_2) (possibly both). No matter which, we can join C_2 to C_1 (as described above) to obtain a circuit D_2 containing every node of Q_1 and Q_2 . (Henceforth, we now treat those links of Q_2 which were temporarily marked as faulty as being healthy again.)

All links of D_2 except for (x_1, x_2) and (y_1, y_2) are links in Q_1 or Q_2 . Hence, there is much potential to join D_2 , as above, to a Hamiltonian circuit in Q_3 or Q_k . Similarly to as before (by applying exactly the same counting argument), w.l.o.g. there exist two consecutive links (u_2, v_2) and (v_2, w_2) of $D_2 \cap Q_2$ such that the links (u_3, v_3) , (v_3, w_3) , (u_2, u_3) , (v_2, v_3) , and (w_2, w_3) are healthy. Again, by arguing exactly as before, there is a Hamiltonian circuit C_3 in Q_3 containing either the link (u_3, v_3) or the link (v_3, w_3) ; and we can join D_2 to C_3 using links in dimension 1 to obtain a circuit D_3 containing all nodes of Q_1 , Q_2 , and Q_3 . Exactly the same arguments apply so that we might extend D_3 to a circuit D_4 , containing all nodes of Q_1 , Q_2 , Q_3 , and Q_4 , and so on until we obtain a Hamiltonian circuit in Q_{k+1}^n .

Case (ii). Each Q_i is such that every node is incident with at least two healthy links and some Q_j has exactly $4n - 4$ faults.

W.l.o.g. we may assume that $j = 1$. Suppose that there is some fault (x_1, y_1) of Q_1 such that (x_1, x_2) and (y_1, y_2) are healthy. Amend Q_1 so that (x_1, y_1) is temporarily marked as healthy. By the induction hypothesis applied to this amended Q_1 , there is a Hamiltonian circuit C_1 which may or may not contain (x_1, y_1) ; and C_1 is a circuit in the original Q_1 . The circuit C_1 has an isomorphic copy C_i in each Q_i for $i = 2, 3, \dots, k$. If (x_1, y_1) is in C_1 , the circuit C_1 can be joined to C_2 using the healthy links (x_1, x_2) and (y_1, y_2) . Otherwise, because there are exactly three faults in dimension 1 and $\lfloor k^n/2 \rfloor > 3$, there is a link (u_1, v_1) of C_1 such that (u_1, u_2) and (v_1, v_2) are healthy. (Use a counting argument similar to that used before except split C_1 into groups of two consecutive vertices and look at the pairs of links in dimension 1 joining Q_1 to Q_2 .) C_1 can now be joined to C_2 using these links to yield a circuit D_2 containing every node of Q_1 and Q_2 . The circuit D_2 contains $k^n - 1$ links of Q_2 . As $\lfloor (k^n - 1)/2 \rfloor > 3$, the same argument yields that there is a link (u_2, v_2) of $D_2 \cap Q_2$ such that the links (w_3, z_3) , (w_2, w_3) , and (z_2, z_3) are all healthy. Moreover, (w_3, z_3) lies on the circuit C_3 of Q_3 . Hence, we can join D_2 and C_3 to obtain a circuit D_3 containing every node of Q_1 , Q_2 , and Q_3 . Exactly the same arguments apply so that we can extend D_3 to a Hamiltonian circuit of Q_{k+1}^n .

On the other hand, suppose that, for every fault (x_1, y_1) of Q_1 , at least one of

(x_1, x_2) and (y_1, y_2) , and at least one of (x_1, x_k) and (y_1, y_k) , are faulty. Let (x_1, y_1) be some fault of Q_1 . As there are exactly three faults in dimension 1, it cannot be the case that two faults in Q_1 are not incident with one another. Let us now count the maximum number μ of faults of Q_1 which could be incident with either x_1 or y_1 . Consider x_1 . The number of faults incident with x_1 , apart from the fault (x_1, y_1) , is at most $2n - 3$. Similarly, the number of faults incident with y_1 , apart from the fault (x_1, y_1) , is at most $2n - 3$. Hence, $\mu \leq (2n - 3) + (2n - 3) + 1 = 4n - 5$. However, there are $4n - 4$ faults in Q_1 and so we obtain a contradiction.

Case (iii). There exists some Q_i in which there is a node incident with exactly one healthy link in Q_i .

W.l.o.g. we may assume that the node x_1 in Q_1 is incident with exactly one healthy link, (x_1, y_1) , in Q_1 . As x_1 is incident with $2n - 1$ faults in Q_1 , each Q_i , for $i = 2, 3, \dots, k$, contains at most $2n - 3$ faults; there is no node in any Q_i , for $i = 2, 3, \dots, k$, which is incident with less than three healthy links in that Q_i ; and apart from x_1 , there is no other node in Q_1 which is incident with less than two healthy links in Q_1 . Also, as x_1 is incident with at least two healthy links in Q_{n+1}^k , we may suppose that (x_1, x_2) is healthy. Consider w_1 , one of the $2n - 1$ potential neighbors of x_1 in Q_1 for which the link (x_1, w_1) is faulty. There are two scenarios.

Case (iii)(a). (w_1, w_2) is a healthy link.

Mark the previously faulty link (x_1, w_1) as temporarily healthy. By the induction hypothesis applied to this amended Q_1 , there is a Hamiltonian path P_1 from x_1 to w_1 . Moreover, this Hamiltonian path P_1 is a Hamiltonian path in the original Q_1 (where the links temporarily marked as faulty resume their healthy status).

Mark some of the previously healthy links in Q_2 that are incident with x_2 as temporarily faulty and mark the link (x_2, w_2) as temporarily healthy (if necessary) so as to ensure that x_2 is incident with exactly two healthy links in this amended Q_2 (one of which is (x_2, w_2)). Note that in order to build this amended Q_2 we have introduced at most $2n - 2$ temporary faults; and so this amended Q_2 has at most $4n - 5$ faults and every node is incident with at least two healthy links. Hence, by the induction hypothesis, there exists a Hamiltonian path P_2 in this amended Q_2 from x_2 to w_2 . Moreover, this Hamiltonian path P_2 is a Hamiltonian path in the original Q_2 . Join P_1 and P_2 using the healthy links (x_1, x_2) and (w_1, w_2) to form a circuit D_2 which contains all nodes of Q_1 and Q_2 .

Applying a counting argument similar to that used in Case (ii), along with the fact that $\lfloor (k^n - 1)/2 \rfloor > 2n$ (note that the total number of faults in Q_{n+1}^k not contained in Q_1 is at most $2n$), there exists a link (u_2, v_2) of $D_2 \cap Q_2$ such that the links (u_3, v_3) , (u_2, u_3) , and (v_2, v_3) are healthy. Temporarily mark healthy links in Q_3 incident with u_3 as faulty so that in this amended Q_3 , u_3 is incident with exactly two healthy links, one of which is (u_3, v_3) . In order to build this amended Q_3 we have introduced at most $2n - 2$ temporary faults; and so this amended Q_3 has at most $4n - 5$ faults and every node is incident with at least two healthy links. By the induction hypothesis, there is a Hamiltonian circuit C_3 in the original Q_3 containing the link (u_3, v_3) . We can join D_2 and C_3 , using the healthy links (u_2, u_3) and (v_2, v_3) , to obtain a circuit D_3 containing every node of Q_1 , Q_2 , and Q_3 . Exactly the same argument can be applied to extend D_3 to a circuit D_4 and so on until we have a Hamiltonian circuit of Q_{n+1}^k .

Case (iii)(b). All links from every such w_1 to its corresponding node w_2 in Q_2 are faulty.

This accounts for another $2n - 1$ faults in Q_{n+1}^k . Also, if (x_1, x_k) is healthy, then

by symmetry we are in Case (iii)(a) (as all but at most one link of the form (w_1, w_k) is healthy). Hence, we may assume that (x_1, x_k) is faulty, and this accounts for all the faults in Q_{n+1}^k .

Consequently, (y_1, y_2) and (y_1, y_k) are both healthy links. (Recall that (x_1, y_1) is the only healthy link of Q_1 incident with x_1 .) Let w_1 be some potential neighbor of x_1 in Q_1 for which the link (x_1, w_1) is faulty. Amend Q_1 by marking the link (x_1, w_1) as temporarily healthy. By the induction hypothesis applied to this amended Q_1 , there is a Hamiltonian path P_1 in the original Q_1 from x_1 to w_1 . Rename the nodes of P_1 as $x_{1,1} = x_1, x_{1,2} = y_1, x_{1,3}, \dots, x_{1,k^n} = w_1$, and note that in each Q_i , $i \geq 2$, there is a corresponding Hamiltonian path P_i which can be extended to a Hamiltonian circuit C_i of Q_i (as (x_i, w_i) is healthy in Q_i). Rename the nodes of C_i as $x_{i,1} = x_i, x_{i,2} = y_i, x_{i,3}, \dots, x_{i,k^n} = w_i$ for each $i \geq 2$.

For ease of notation, denote k^n by m . Suppose k is even. Then the following is a Hamiltonian circuit in Q_{n+1}^k :

$$\begin{aligned} & (x_{1,1}, x_{2,1}, \dots, x_{k,1}, x_{k,2}, x_{k,3}, x_{1,3}, x_{1,4}, \dots, x_{1,m}, x_{k,m}, x_{k-1,m}, \dots, x_{2,m}, \\ & x_{2,m-1}, x_{3,m-1}, \dots, x_{k,m-1}, x_{k,m-2}, x_{k-1,m-2}, \dots, x_{2,m-2}, x_{2,m-3}, \\ & x_{3,m-3}, \dots, x_{k,m-3}, x_{k,m-4}, \dots, x_{k,4}, x_{k-1,4}, \dots, x_{2,4}, x_{2,3}, x_{3,3}, \dots, \\ & x_{k-1,3}, x_{k-1,2}, x_{k-2,2}, \dots, x_{2,2}, x_{1,2}, x_{1,1}). \end{aligned}$$

(See Figure 2.2 where some of the healthy links between the Q_i 's are shown and bold links denote the links of the Hamiltonian circuit.) If k is odd, then the following is a Hamiltonian circuit in Q_{n+1}^k :

$$\begin{aligned} & (x_{1,1}, x_{2,1}, \dots, x_{k,1}, x_{k,2}, x_{k-1,2}, \dots, x_{2,2}, x_{2,3}, x_{3,3}, \dots, x_{k,3}, x_{k,4}, x_{k-1,4}, \dots, \\ & x_{2,4}, x_{2,5}, \dots, x_{2,m}, x_{3,m}, \dots, x_{2,m}, x_{k,m}, \dots, x_{1,m}, x_{1,m-1}, \dots, x_{1,2}, x_{1,1}). \end{aligned}$$

(See Figure 2.3.)

Case (iv). There exists some Q_i in which there is a node incident with no healthy links in Q_i .

W.l.o.g. we may assume that x_1 is incident with no healthy links in Q_1 . As x_1 is incident with at least two healthy links in Q_{n+1}^k , the links (x_1, x_2) and (x_1, x_k) must be healthy. There are at least $2n$ faults in Q_1 , and so there must be at most $2n - 4$ faults distributed amongst Q_2, Q_3, \dots, Q_k . Hence, apart from x_1 , there are no nodes which are incident with less than four healthy links in their respective copy of Q_n^k .

The node x_1 has $2n$ potential neighbors in Q_1 . Each of these potential neighbors is incident with a potential dimension 1 link to Q_1 and a potential dimension 1 link to Q_k . (These dimension 1 links might be faulty.) As there are at most $2n - 1$ faults in dimension 1, there must exist potential neighbors y_1 and z_1 of x_1 such that the links (y_1, y_2) and (z_1, z_k) are healthy. (Partition the potential neighbors into n pairs $\{y_1, z_1\}$ and look at the pairs of dimension 1 links $\{(y_1, y_2), (z_1, z_k)\}$ and $\{(y_1, y_k), (z_1, z_2)\}$.) Mark the faulty links (x_1, y_1) and (x_1, z_1) as temporarily healthy in Q_1 . Applying the induction hypothesis to this amended Q_1 , we obtain a path P_1 in the original Q_1 from y_1 to z_1 upon which every node of Q_1 appears exactly once, except for x_1 which does not appear at all.

By marking previously healthy links in Q_2 that are incident with x_2 as temporarily faulty, and by marking the link (x_2, y_2) as temporarily healthy (if necessary), ensure that x_2 is incident with exactly two healthy links in this amended Q_2 , one of which is (x_2, y_2) . This involves introducing at most $2n - 2$ temporary faults into Q_2 ; and so the amended Q_2 has at most $4n - 6$ faults and every node is incident with at least

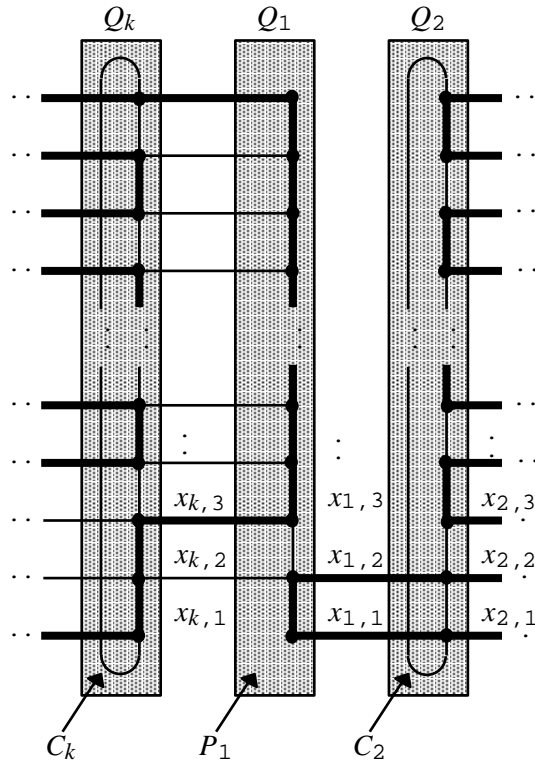


FIG. 2.2. The Hamiltonian circuit when k is even.

two healthy links. The induction hypothesis yields that there is a Hamiltonian path from x_2 to y_2 in the original Q_2 . Likewise, there is a Hamiltonian path from x_k to z_k in Q_k . Hence, let D_2 be the circuit obtained by joining P_1 , P_2 , and P_k using the healthy links (x_1, x_2) , (y_1, y_2) , (x_1, x_k) , and (z_1, z_k) .

Applying a counting argument similar to that used in Case (ii), along with the fact that $\lfloor (k^n - 1)/2 \rfloor > 2n - 1$ (note that the total number of faults in Q_{n+1}^k not contained in Q_1 is at most $2n - 1$), there exists a link (u_2, v_2) of $D_2 \cap Q_2$ such that the links (u_3, v_3) , (u_2, u_3) , and (v_2, v_3) are healthy. Temporarily mark healthy links in Q_3 incident with u_3 as faulty so that in this amended Q_3 , u_3 is incident with exactly two healthy links, one of which is (u_3, v_3) . In order to build this amended Q_3 we have introduced at most $2n - 2$ temporary faults; and so this amended Q_3 has at most $4n - 6$ faults and every node is incident with at least two healthy links. By the induction hypothesis, there is a Hamiltonian circuit C_3 in the original Q_3 containing the link (u_3, v_3) . We can join D_2 and C_3 , using the healthy links (u_2, u_3) and (v_2, v_3) , to obtain a circuit D_3 containing every node of Q_k , Q_1 , Q_2 , and Q_3 . Exactly the same argument can be applied to extend D_3 to a circuit D_4 and so on until we have a Hamiltonian circuit of Q_{n+1}^k .

It remains to show that the result holds for the base cases of the induction, namely, when $n = 2$ and $k \geq 4$, and when $n = 3$ and $k = 3$.

LEMMA 2.2. *If Q_2^k , where $k \geq 4$, has three faulty links and is such that every node is incident with at least two healthy links, then Q_2^k has a Hamiltonian circuit.*

Proof. There exists some dimension, say dimension 1, that contains at least two

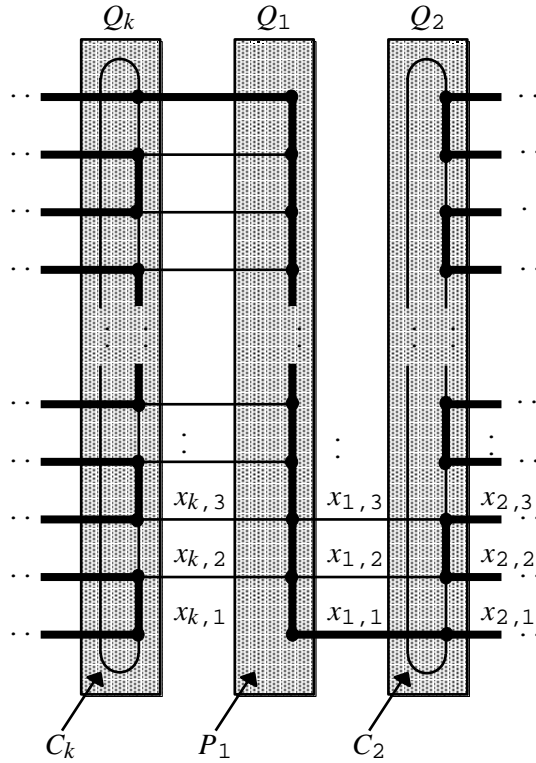


FIG. 2.3. The Hamiltonian circuit when k is odd.

faults. Partition Q_2^k over dimension 1 to obtain k copies of Q_1^k , namely Q_1, Q_2, \dots, Q_k .

Case (i). All faults are in dimension 1.

Consider the circuit Q_1 of length k . As there are three faults in dimension 1, w.l.o.g. there exists an edge (x_1, y_1) of Q_1 such that the links (x_1, x_2) and (y_1, y_2) are both healthy. (Apply our usual counting argument.) Join Q_1 and Q_2 using these links to obtain a circuit D_2 containing every node of Q_1 and Q_2 . By proceeding as we have done throughout, the same argument can be used to extend D_2 to (w.l.o.g.) a circuit D_3 and so on until we obtain a Hamiltonian circuit of Q_2^k .

Case (ii). Dimension 1 has exactly two faults.

W.l.o.g. the only fault not in dimension 1 may be assumed to be (x_1, y_1) in Q_1 . If the links (x_1, x_2) and (y_1, y_2) are both healthy or the links (x_1, x_k) and (y_1, y_k) are both healthy, then we can join Q_1 with Q_2 or Q_k , respectively, as in Case (i), and extend this circuit to a Hamiltonian circuit of Q_2^k .

Hence, w.l.o.g. we may assume that the links (x_1, x_2) and (y_1, y_k) are both faulty. If k is even, then there exists a Hamiltonian circuit in Q_2^k as pictured in Figure 2.2. (In that picture, $x_{1,3}, x_{1,2}, x_{2,3}$, and $x_{k,2}$ play the roles of x_1, y_1, x_2 , and y_k , respectively.) If k is odd, then there exists a Hamiltonian circuit in Q_2^k as pictured in Figure 2.3. (In that picture, $x_{1,m}, x_{1,1}, x_{2,m}$, and $x_{k,1}$ play the roles of x_1, y_1, x_2 , and y_k , respectively.) \square

LEMMA 2.3. If Q_2^3 has three faulty links and is such that every node is incident with at least two healthy links, then Q_2^3 has a Hamiltonian circuit unless these three faulty links form a circuit of length 3.

Proof. There exists some dimension, say dimension 1, that contains at least two faults. Partition Q_2^3 over dimension 1 to obtain three copies of Q_1^3 , namely $Q_1, Q_2,$ and Q_3 . We may assume that either Q_1 contains one fault or all faults are in dimension 1. Denote the nodes of Q_i by $x_i, y_i,$ and z_i for $i = 1, 2, 3$.

Case (i). Q_1 contains one fault.

W.l.o.g. we may assume that the fault in Q_1 is (x_1, y_1) .

Case (i)(a). The links (x_1, x_2) and (y_1, y_2) are healthy.

Form the circuit $C = (x_1, z_1, y_1, y_2, z_2, x_2, x_1)$ in Q_2^3 . There are two possibilities: either one of the sets of pairs

$$\{(x_1, x_3), (z_1, z_3)\}, \{(y_1, y_3), (z_1, z_3)\}, \{(x_2, x_3), (z_2, z_3)\}, \{(y_2, y_3), (z_2, z_3)\}$$

consists of two healthy links or the faulty links in dimension 1 are (z_1, z_3) and (z_2, z_3) . In the former case, the circuit C can be joined to the circuit (x_3, y_3, z_3, x_3) using the pair of healthy links to obtain a Hamiltonian circuit in Q_2^3 : in the latter case, we can define our Hamiltonian circuit in Q_2^3 to be $(x_1, z_1, z_2, y_2, y_1, y_3, z_3, x_3, x_2, x_1)$.

Case (i)(b). At least one of the links (x_1, x_2) and (y_1, y_2) is faulty.

By symmetry, we may also assume that at least one of (x_1, x_3) and (y_1, y_3) is faulty (as otherwise we are in Case (i)(a)); so this accounts for all faults in Q_2^3 . The only configuration possible, up to isomorphism, is that in Figure 2.4(a), and so there is a Hamiltonian circuit as depicted in that figure. (In Figure 2.4(a), the nodes $x_1, y_1,$ and z_1 of Q_1 form the central column, with the other two columns similarly depicting the nodes of Q_2 and Q_3 . Faults are denoted by missing links, and links of the Hamiltonian circuit are drawn in bold.)

Case (ii). All faults are in dimension 1.

Up to isomorphism, there are six different configurations possible, shown in Figure 2.4(b)–(g), with Hamiltonian circuits as depicted except for Figure 2.4(g) where no such Hamiltonian circuit exists. (In Figure 2.4(g), w.l.o.g. the bold links are necessarily in any Hamiltonian circuit, if there were to exist one; and one can immediately see that there is no extension of these bold links to a Hamiltonian circuit.) \square

LEMMA 2.4. *If Q_3^3 has seven faulty links and is such that every node is incident with at least two healthy links, then Q_3^3 has a Hamiltonian circuit.*

Proof. Case (i). Q_3^3 contains faults forming a circuit C of length 3.

All of the faults in C must appear in the same dimension, say dimension 1. Partition Q_3^3 across dimension 1 to obtain three copies of Q_1^3 , namely $Q_1, Q_2,$ and Q_3 , and let the faulty links in C be $(x_1, x_2), (x_2, x_3),$ and (x_3, x_1) . We may assume that Q_1 contains the most faults amongst these copies, then $Q_2,$ and then Q_3 .

Case (i)(a). Q_1 contains faults forming a circuit D of length 3.

Let y_1 and z_1 be nodes of D different from x_1 (x_1 may or may not be on D) so that the number of faults incident with y_1 is no greater than the number of faults incident with any node of D different from x_1 . (Note that x_1 is incident with at most two faults in Q_1 .) If y_1 is incident with one healthy link in Q_1 , then every other node of Q_1 is incident with at least two healthy links in Q_1 . (As Q_3^3 has seven faults, y_1 must be incident with at least one healthy link in Q_1 .) In this case, temporarily mark the link (y_1, z_1) as healthy so that there are at most three faults in the amended Q_1 . (And these faults do not form a circuit.) Lemma 2.3 yields that there is a Hamiltonian path in the original Q_1 from y_1 to z_1 .

If y_1 is incident with two healthy links in Q_1 , then every node in Q_1 is incident with at least two healthy links in Q_1 . Mark the link (y_1, z_1) as temporarily healthy and a healthy link of Q_1 incident with y_1 as temporarily faulty. Every node in the

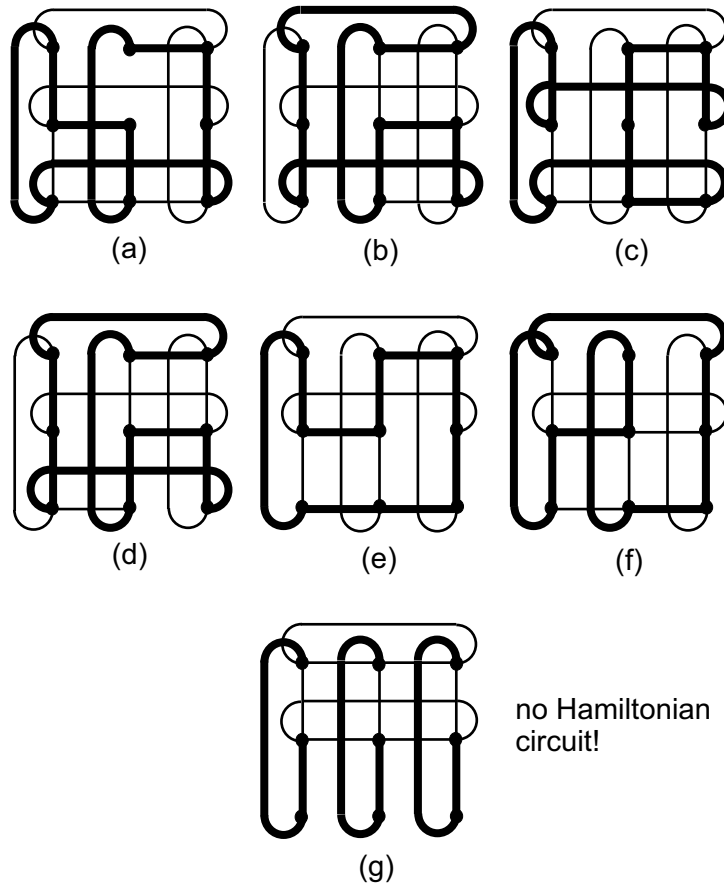


FIG. 2.4. The different configurations for Q_2^3 .

amended Q_1 is incident with at least two healthy links, and there are at most three faults. (And these faults do not form a circuit.) Lemma 2.3 yields that there is a Hamiltonian path in the original Q_1 from y_1 to z_1 .

Whichever of the above scenarios applies, denote the Hamiltonian path in Q_1 from y_1 to z_1 by P_1 . The faults in Q_1 and the faults (x_1, x_2) , (x_2, x_3) , and (x_3, x_1) account for at least six of the seven faults in Q_3^3 . Hence, w.l.o.g. we may assume that the links (x_1, x_2) and (y_1, y_2) are healthy. There is at most one fault in Q_2 . By marking healthy links of Q_2 as temporarily faulty (if necessary), ensure that (y_2, z_2) is healthy and y_2 is incident with exactly two healthy links. Applying Lemma 2.3 to this amended Q_2 yields that there is a Hamiltonian circuit C_2 (that is also a Hamiltonian circuit in the original Q_2) including the link (y_2, z_2) . Join P_1 and C_2 using the healthy links (y_1, y_2) and (z_1, z_2) to obtain a circuit D_2 containing every node of Q_1 and Q_2 .

Q_3 has an isomorphic copy C_3 of C_2 , and there are no faults in Q_3 . As C_3 has length 9 and there are at most four faults in dimension 1, by applying our counting argument as we have done throughout, we can join D_2 and C_3 using appropriate dimension 1 links to obtain a Hamiltonian circuit in Q_3^3 .

Case (i)(b). Q_1 does not contain faults forming a circuit D of length 3.

Note that the proofs of Cases (i), (ii), (iii), and (iv) of the main theorem hold for

Q_3^3 except that, throughout, instead of appealing to an inductive hypothesis, we use Lemma 2.3; in Case (i), we assume that dimension 1 contains at most five faults; and in Case (iii)(a), when amending Q_2 we must ensure that we do not introduce a circuit of faults of length 3. (This can be done as Q_2 has at most 1 fault.) Consequently, we are left with one scenario to consider: the subcase of Case (i) when each Q_i is such that every node is incident with at least two healthy links and when dimension 1 contains six or seven faults.

Let (a new) 3-ary 2-cube Q_2^3 be such that there is a fault (x, y) in Q_2^3 if and only if there is a fault (x_i, y_i) in Q_i for some $i \in \{1, 2, 3\}$. Then Q_2^3 has at most two faults and, by Lemma 2.3, it has a Hamiltonian circuit C . For each $i \in \{1, 2, 3\}$, let C_i be the isomorphic copy of C in Q_i . (Note that each C_i consists entirely of healthy links.) Even if dimension 1 (of our original Q_3^3) contains seven faults, our usual counting argument yields that there exists a pair of healthy links $\{(u_1, u_2), (v_1, v_2)\}$ or $\{(u_1, u_3), (v_1, v_3)\}$, where (u_1, v_1) is a link of C_1 : w.l.o.g. we may assume that these healthy links are (u_1, u_2) and (v_1, v_2) . We can join C_1 and C_2 using these healthy links and then proceed similarly to join the resulting circuit to C_3 and obtain a Hamiltonian circuit of Q_3^3 .

Case (ii). Q_3^3 does not contain faults forming a circuit of length 3.

There exists a dimension, say dimension 1, containing at least three faults. Partition Q_3^3 across dimension 1 to obtain three copies of Q_2^3 , namely $Q_1, Q_2,$ and Q_3 . Let Q_1 contain the most faults amongst these copies, then Q_2 , and then Q_3 . Proceeding as in Case (i)(b) yields the result. \square

The main theorem now follows by induction. \square

The result in Theorem 2.1 is optimal in the following sense. Let $a, b, c,$ and d be four nodes in Q_n^k , where $k \geq 4$ and $n \geq 2$, or $k = 3$ and $n \geq 3$, such that there are links $(a, b), (b, c), (c, d),$ and (d, a) . Let the faults of Q_n^k consist of those links incident with a that are different from (a, b) and (a, d) , and those links incident with c that are different from (b, c) and (c, d) . In particular, Q_n^k has $4n - 4$ faults and every node is incident with at least two healthy links; but this faulty Q_n^k does not contain a Hamiltonian circuit, as any Hamiltonian circuit necessarily contains the links (a, b) and (a, d) , and also the links (c, b) and (c, d) , which yields a contradiction.

3. Conclusions. We have proven that every k -ary n -cube Q_n^k which has at most $4n - 5$ faulty links and is such that every node is incident with at least two healthy links has a Hamiltonian circuit. As mentioned earlier, an analogous result for hypercubes was proven by Chan and Lee [8]. In [8], it was also shown that the problem of deciding whether a faulty binary n -cube has a Hamiltonian circuit is NP-complete. Their complexity-theoretic reduction (from the 3-satisfiability problem) can easily be adapted to show that the problem of deciding whether a faulty k -ary n -cube has a Hamiltonian circuit is also NP-complete. (We leave the proof of this as a simple exercise.)

As open problems relating to the research in this paper, we propose the following. The construction of our Hamiltonian circuits in our faulty k -ary n -cubes does not yield efficient parallel distributed algorithms for actually building the Hamiltonian circuits. For example, suppose one had a parallel computer whose underlying interconnection network was a k -ary n -cube and each node, i.e., processor, had local (or even global) knowledge of the faulty links. How could we develop an efficient message-passing algorithm so that, upon termination, every node knew its successor and predecessor on a Hamiltonian circuit (without necessarily knowing the Hamiltonian circuit in its entirety)? Such an algorithm would be extremely useful. Also, whilst we provide a

precise result as to the threshold value on the number of faulty links occurring in a k -ary n -cube so that there still exists a Hamiltonian circuit (under the assumption that every node is incident with at least two healthy links) and we also remark that the general decision problem is NP-complete, it would be useful if “safe patterns” of faults could be established so that even though there were more than $4n - 5$ faulty links present, one could still be sure of the existence of a Hamiltonian circuit because these faults were arranged in some specific formation. Finally, we have addressed only the problem of finding longest circuits in k -ary n -cubes in the presence of faulty links. It would be interesting to do likewise in the presence of faulty nodes, or even faulty nodes and links.

REFERENCES

- [1] Y.A. ASHIR AND I.A. STEWART, *On embedding cycles in k -ary n -cubes*, *Parallel Process. Lett.*, 7 (1997), pp. 49–55.
- [2] Y.A. ASHIR, I.A. STEWART, AND A. AHMED, *Communication algorithms in k -ary n -cube interconnection networks*, *Inform. Process. Lett.*, 61 (1997), pp. 43–48.
- [3] D. BERTSEKAS AND J. TSITSIKLIS, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [4] S. BETTAYEB, *On the k -ary hypercube*, *Theoret. Comput. Sci.*, 140 (1995), pp. 333–339.
- [5] L. BHUYAN AND D. AGRAWAL, *Generalized hypercube and hyperbus structures for a computer network*, *IEEE Trans. Comput.*, 33 (1984), pp. 323–333.
- [6] B. BOSE, B. BROEG, Y. KWON, AND Y. ASHIR, *Lee distance and topological properties of k -ary n -cubes*, *IEEE Trans. Comput.*, 44 (1995), pp. 1021–1030.
- [7] R.B. BROEG, *Topics in Toroidal Interconnection Networks*, Ph.D. thesis, Oregon State University, Corvallis, OR, 1995.
- [8] M.Y. CHAN AND S.-J. LEE, *On the existence of Hamiltonian circuits in faulty hypercubes*, *SIAM J. Discrete Math.*, 4 (1991), pp. 511–527.
- [9] W. DALLY, *Performance analysis of k -ary n -cube interconnection networks*, *IEEE Trans. Comput.*, 39 (1990), pp. 775–785.
- [10] F.T. LEIGHTON, *Introduction to Parallel Algorithms and Architectures: Arrays. Trees. Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.
- [11] D. LINDER AND J. HARDEN, *An adaptive and fault tolerant wormhole routing strategy for k -ary n -cubes*, *IEEE Trans. Comput.*, 40 (1991), pp. 2–12.
- [12] Y. SAAD AND M. SCHULTZ, *Data communications in hypercubes*, *J. Parallel Distrib. Comput.*, 6 (1989), pp. 115–135.

IDEAL BINARY CLUTTERS, CONNECTIVITY, AND A CONJECTURE OF SEYMOUR*

GÉRARD CORNUÉJOLS[†] AND BERTRAND GUENIN[‡]

Abstract. A binary clutter is the family of odd circuits of a binary matroid, that is, the family of circuits that intersect with odd cardinality a fixed given subset of elements. Let A denote the $0, 1$ matrix whose rows are the characteristic vectors of the odd circuits. A binary clutter is ideal if the polyhedron $\{x \geq \mathbf{0} : Ax \geq \mathbf{1}\}$ is integral. Examples of ideal binary clutters are st -paths, st -cuts, T -joins or T -cuts in graphs, and odd circuits in weakly bipartite graphs. In 1977, Seymour [*J. Combin. Theory Ser. B*, 22 (1977), pp. 289–295] conjectured that a binary clutter is ideal if and only if it does not contain \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , or $b(\mathcal{O}_{K_5})$ as a minor. In this paper, we show that a binary clutter is ideal if it does not contain five specified minors, namely the three above minors plus two others. This generalizes Guenin’s characterization of weakly bipartite graphs [*J. Combin. Theory Ser. B*, 83 (2001), pp. 112–168], as well as the theorem of Edmonds and Johnson [*Math. Programming*, 5 (1973), pp. 88–124] on T -joins and T -cuts.

Key words. ideal clutter, signed matroid, multicommodity flow, weakly bipartite graph, T -cut, Seymour’s conjecture, connectivity, separation

AMS subject classifications. 90C10, 90C27, 52B40

PII. S0895480100371389

1. Introduction. A clutter \mathcal{H} is a finite family of sets, over some finite ground set $E(\mathcal{H})$, with the property that no set of \mathcal{H} contains, or is equal to, another set of \mathcal{H} . A clutter is said to be *ideal* if the polyhedron $\{x \in \mathbb{R}_+^{E(\mathcal{H})} : \sum_{i \in S} x_i \geq 1 \text{ for all } S \in \mathcal{H}\}$ is an integral polyhedron; that is, all its extreme points have $0, 1$ coordinates. A clutter \mathcal{H} is *trivial* if $\mathcal{H} = \emptyset$ or $\mathcal{H} = \{\emptyset\}$. Given a nontrivial clutter \mathcal{H} , we write $A(\mathcal{H})$ for a $0, 1$ matrix whose columns are indexed by $E(\mathcal{H})$ and whose rows are the characteristic vectors of the sets $S \in \mathcal{H}$. With this notation, a nontrivial clutter \mathcal{H} is ideal if and only if $\{x \geq \mathbf{0} : A(\mathcal{H})x \geq \mathbf{1}\}$ is an integral polyhedron.

Given a clutter \mathcal{H} , a set $T \subseteq E(\mathcal{H})$ is a *transversal* of \mathcal{H} if T intersects all the members of \mathcal{H} . The clutter $b(\mathcal{H})$, called the *blocker* of \mathcal{H} , is defined as follows: $E(b(\mathcal{H})) = E(\mathcal{H})$ and $b(\mathcal{H})$ is the set of inclusion-wise minimal transversals of \mathcal{H} . It is well known that $b(b(\mathcal{H})) = \mathcal{H}$ [13]. Hence we say that $\mathcal{H}, b(\mathcal{H})$ form a *blocking pair* of clutters. Lehman [14] showed that if a clutter is ideal, then so is its blocker. A clutter is said to be *binary* if, for any $S_1, S_2, S_3 \in \mathcal{H}$, their symmetric difference $S_1 \triangle S_2 \triangle S_3$ contains, or is equal to, a set of \mathcal{H} .

Given a clutter \mathcal{H} and $i \in E(\mathcal{H})$, the *contraction* \mathcal{H}/i and *deletion* $\mathcal{H} \setminus i$ are clutters defined as follows: $E(\mathcal{H}/i) = E(\mathcal{H} \setminus i) = E(\mathcal{H}) - \{i\}$, the family \mathcal{H}/i is the set of inclusion-wise minimal members of $\{S - \{i\} : S \in \mathcal{H}\}$, and $\mathcal{H} \setminus i = \{S : i \notin S \in \mathcal{H}\}$. Contractions and deletions can be performed sequentially, and the result does not depend on the order. A clutter obtained from \mathcal{H} by a set of deletions J_d and a set of contractions J_c (where $J_c \cap J_d = \emptyset$) is called a *minor* of \mathcal{H} and is denoted by $\mathcal{H} \setminus J_d / J_c$.

*Received by the editors April 21, 2000; accepted for publication March 12, 2002; published electronically May 8, 2002. This work was supported in part by NSF grants DMI-0098427, DMI-9802773, DMS 96-32032, and DMS-9509581, and ONR grant N00014-9710196.

<http://www.siam.org/journals/sidma/15-3/37138.html>

[†]Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 (gc0v@andrew.cmu.edu).

[‡]Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada (bguenin@math.uwaterloo.ca).

It is a *proper* minor if $J_c \cup J_d \neq \emptyset$. A clutter is said to be *minimally nonideal* (mni) if it is not ideal but all its proper minors are ideal.

The clutter \mathcal{O}_{K_5} is defined as follows: $E(\mathcal{O}_{K_5})$ is the set of 10 edges of the complete graph K_5 and \mathcal{O}_{K_5} is the set of odd circuits of K_5 (the triangles and the circuits of length 5). The 10 constraints corresponding to the triangles define a fractional extreme point $(\frac{1}{3}, \frac{1}{3}, \dots, \frac{1}{3})$ of the associated polyhedron $\{x \geq \mathbf{0} : A(\mathcal{O}_{K_5})x \geq \mathbf{1}\}$. Thus \mathcal{O}_{K_5} is not ideal and neither is its blocker. The clutter \mathcal{L}_{F_7} is the family of circuits of length three of the Fano matroid (or, equivalently, the family of lines of the Fano plane), i.e., $E(\mathcal{L}_{F_7}) = \{1, 2, 3, 4, 5, 6, 7\}$ and

$$\mathcal{L}_{F_7} = \{\{1, 3, 5\}, \{1, 4, 6\}, \{2, 3, 6\}, \{2, 4, 5\}, \{1, 2, 7\}, \{3, 4, 7\}, \{5, 6, 7\}\}.$$

The fractional point $(\frac{1}{3}, \frac{1}{3}, \dots, \frac{1}{3})$ is an extreme point of the associated polyhedron, and hence \mathcal{L}_{F_7} is not ideal. The blocker of \mathcal{L}_{F_7} is \mathcal{L}_{F_7} itself. The following excluded minor characterization is predicted.

Seymour’s conjecture (Seymour [23, p. 200], [26, (9.2), (11.2)]). A binary clutter is ideal if and only if it has no \mathcal{L}_{F_7} , no \mathcal{O}_{K_5} , and no $b(\mathcal{O}_{K_5})$ minor.

Consider a clutter \mathcal{H} and an arbitrary element $t \notin E(\mathcal{H})$. We write \mathcal{H}^+ for the clutter with $E(\mathcal{H}^+) = E(\mathcal{H}) \cup \{t\}$ and $\mathcal{H}^+ = \{S \cup \{t\} : S \in \mathcal{H}\}$. The clutter Q_6 is defined as follows: $E(Q_6)$ is the set of edges of the complete graph K_4 and Q_6 is the set of triangles of K_4 . The clutter Q_7 is defined as follows:

$$A(Q_7) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Note that the first six columns of $A(Q_7)$ form the matrix $A(b(Q_6))$.

The main result of this paper is that Seymour’s conjecture holds for the class of clutters that do not have Q_6^+ and Q_7^+ minors.

THEOREM 1.1. *A binary clutter is ideal if it does not have \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , $b(\mathcal{O}_{K_5})$, Q_6^+ , or Q_7^+ as a minor.*

Since the blocker of an ideal binary clutter is also an ideal, we can restate Theorem 1.1 as follows.

COROLLARY 1.2. *A binary clutter is ideal if it does not have \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , $b(\mathcal{O}_{K_5})$, $b(Q_7^+)$, or $b(Q_6^+)$ as a minor.*

We say that \mathcal{H} is the *clutter of odd circuits* of a graph G if $E(\mathcal{H})$ is the set of edges of G and \mathcal{H} is the set of odd circuits of G . A graph is said to be *weakly bipartite* if the clutter of its odd circuits is ideal. This class of graphs has a nice excluded minor characterization.

THEOREM 1.3 (Guenin [10]). *A graph is weakly bipartite if and only if its clutter of odd circuits has no \mathcal{O}_{K_5} minor.*

The class of clutters of odd circuits is closed under minor taking (Remark 8.2). Moreover, one can easily check that \mathcal{O}_{K_5} is the only clutter of odd circuits among the five excluded minors of Theorem 1.1 (see Remark 8.3 and [20]). It follows that Theorem 1.1 implies Theorem 1.3. It does not provide a new proof of Theorem 1.3, however, as we shall use Theorem 1.3 to prove Theorem 1.1.

Consider a graph G and a subset T of its vertices of even cardinality. A T -join is an inclusion-wise minimal set of edges J such that T is the set of vertices of odd degree of the edge-induced subgraph $G[J]$. A T -cut is an inclusion-wise minimal set of edges $\delta(U) := \{(u, v) : u \in U, v \notin U\}$, where U is a set of vertices of G that satisfies $|U \cap T|$ odd. T -joins and T -cuts generalize many interesting special cases. If $T = \{s, t\}$, then the T -joins (resp., T -cuts) are the st -paths (resp., inclusion-wise minimal st -cuts) of G . If $T = V$, then the T -joins of size $|V|/2$ are the perfect matchings of G . The case where T is identical to the set of odd-degree vertices of G is known as the Chinese postman problem [6, 12]. The families of T -joins and T -cuts form a blocking pair of clutters.

THEOREM 1.4 (Edmonds and Johnson [6]). *The clutters of T -cuts and T -joins are ideal.*

The class of clutters of T -cuts is closed under minor taking (Remark 8.2). Moreover, it is not hard to check that none of the five excluded minors of Theorem 1.1 are clutters of T -cuts (see Remark 8.3 and [20]). Thus Theorem 1.1 implies that the clutter of T -cuts is ideal and thus that its blocker, the clutter of T -joins, is ideal. Hence Theorem 1.1 implies Theorem 1.4. However, we shall also rely on this result to prove Theorem 1.1.

The paper is organized as follows. Section 2 considers representations of binary clutters in terms of signed matroids and matroid ports. Section 3 reviews the notions of lifts and sources, which are families of binary clutters associated with a given binary matroid [20, 29]. Connections between multicommodity flows and ideal clutters are discussed in section 4. The material presented in sections 2, 3, and 4 is not all new. We present it here for the sake of completeness and in order to have a unified framework for the remainder of the paper. In sections 5, 6, and 7 we show that mni clutters do not have small separations. The proof of Theorem 1.1 is given in section 8. Finally, section 9 presents an intriguing example of an ideal binary clutter.

2. Binary matroids and binary clutters. We assume that the reader is familiar with the basics of matroid theory. For an introduction and all undefined terms, see, for instance, Oxley [21]. Given a matroid M , the set of its elements is denoted by $E(M)$ and the set of its circuits by $\Omega(M)$. The dual of M is written M^* . The *deletion* minor $M \setminus e$ of M is the matroid defined as follows: $E(M \setminus e) = E(M) - \{e\}$ and $\Omega(M \setminus e) = \{C : e \notin C \in \Omega(M)\}$. The *contraction* minor M/e of M is defined as $(M^* \setminus e)^*$. Contractions and deletions can be performed sequentially, and the result does not depend on the order. A matroid obtained from M by a set of deletions J_d and a set of contractions J_c is a *minor* of M and is denoted by $M \setminus J_d / J_c$.

A matroid M is *binary* if there exists a 0, 1 matrix A with column set $E(M)$ such that the independent sets of M correspond to independent sets of columns of A over the two element field. We say that A is a *representation* of M . Equivalently, a 0, 1 matrix A is a representation of a binary matroid M if the rows of A span the circuit space of M^* . If C_1 and C_2 are two cycles of a binary matroid, then $C_1 \triangle C_2$ is also a cycle of M . In particular, this implies that every cycle of M can be partitioned into circuits. Let M be a binary matroid and $\Sigma \subseteq E(M)$. The pair (M, Σ) is called a *signed matroid*, and Σ is called the *signature* of M . We say that a circuit C of M is *odd* (resp., *even*) if $|C \cap \Sigma|$ is odd (resp., even).

The results in this section are fairly straightforward and have appeared explicitly or implicitly in the literature [8, 13, 20, 23]. We include some of the proofs for the sake of completeness.

PROPOSITION 2.1 (Lehman [13]). *The following statements are equivalent for a*

clutter: (i) \mathcal{H} is binary; (ii) for every $S \in \mathcal{H}$ and $T \in b(\mathcal{H})$, $|S \cap T|$ is odd; (iii) for every $S_1, \dots, S_k \in \mathcal{H}$ where k is odd, $S_1 \Delta S_2 \Delta \dots S_k$ contains, or is equal to, an element of \mathcal{H} .

PROPOSITION 2.2. *The odd circuits of a signed matroid (M, Σ) form a binary clutter.*

Proof. Let C_1, C_2, C_3 be three odd circuits of (M, Σ) . Then $L := C_1 \Delta C_2 \Delta C_3$ is a cycle of M . Since each of C_1, C_2, C_3 intersects Σ with odd parity, so does L . Since M is binary, L can be partitioned into a family of circuits. One of these circuits must be odd since $|L \cap \Sigma|$ is odd. The result now follows from the definition of binary clutters (see section 1). \square

PROPOSITION 2.3. *Let \mathcal{F} be a clutter such that $\emptyset \notin \mathcal{F}$. Consider the following properties: (i) For all $C_1, C_2 \in \mathcal{F}$ and $e \in C_1 \cap C_2$ there exists $C_3 \in \mathcal{F}$ such that $C_3 \subseteq C_1 \cup C_2 - \{e\}$. (ii) For all $C_1, C_2 \in \mathcal{F}$ there exists $C_3 \in \mathcal{F}$ such that $C_3 \subseteq C_1 \Delta C_2$. If property (i) holds, then \mathcal{F} is the set of circuits of a matroid. If property (ii) holds, then \mathcal{F} is the set of circuits of a binary matroid.*

Property (i) is known as the *circuit elimination axiom*. Circuits of matroids satisfy this property. Note that property (ii) implies property (i). Both results are standard; see Oxley [21].

PROPOSITION 2.4. *Let \mathcal{H} be a binary clutter such that $\emptyset \notin \mathcal{H}$. Let \mathcal{F} be the clutter consisting of all inclusion-wise minimal, nonempty sets obtained by taking the symmetric difference of an arbitrary number of sets of \mathcal{H} . Then $\mathcal{H} \subseteq \mathcal{F}$ and \mathcal{F} is the set of circuits of a binary matroid.*

Proof. By definition, \mathcal{F} satisfies property (ii) in Proposition 2.3. Thus \mathcal{F} is the set of circuits of a binary matroid M . Suppose for a contradiction there is $S \in \mathcal{H} - \mathcal{F}$. Then there exists $S' \in \mathcal{F}$ such that $S' \subset S$. Thus S' is the symmetric difference of a family of, say t , sets of \mathcal{H} . If t is odd, then Proposition 2.1 implies that S' contains a set of \mathcal{H} . If t is even, then Proposition 2.1 implies that $S' \Delta S$ contains a set of \mathcal{H} . Thus S is not inclusion-wise minimal, a contradiction. \square

Consider a binary clutter \mathcal{H} such that $\emptyset \notin \mathcal{H}$. The matroid defined in Proposition 2.4 is called the *up matroid* and is denoted by $u(\mathcal{H})$. Proposition 2.1 implies that every circuit of $u(\mathcal{H})$ is either an element of \mathcal{H} or the symmetric difference of an even number of sets of \mathcal{H} . Since \mathcal{H} is a binary clutter, sets of $b(\mathcal{H})$ intersect with odd parity the circuits of $u(\mathcal{H})$ that are elements of \mathcal{H} . Hence, we have the following remark.

Remark 2.5. A binary clutter \mathcal{H} such that $\emptyset \notin \mathcal{H}$ is the clutter of odd circuits of $(u(\mathcal{H}), \Sigma)$, where $\Sigma \in b(\mathcal{H})$.

Moreover, this representation is essentially unique.

PROPOSITION 2.6. *Let \mathcal{H} be the clutter of odd circuits of the signed matroid (M, Σ) . If \mathcal{H} is not trivial and N is connected, then $N = u(\mathcal{H})$.*

To prove this, we use the following result (see Oxley [21, Theorem 4.3.2]).

THEOREM 2.7 (Lehman [13]). *Let e be an element of a connected binary matroid M . The circuits of M not containing e are of the form $C_1 \Delta C_2$, where C_1 and C_2 are circuits of M containing e .*

We shall also need the following observation which follows directly from Proposition 2.3.

PROPOSITION 2.8. *Let (M, Σ) be a signed matroid and e an element not in $E(M)$. Let $\mathcal{F} := \{C \cup \{e\} : C \in \Omega(M), |C \cap \Sigma| \text{ odd}\} \cup \{C : C \in \Omega(M), |C \cap \Sigma| \text{ even}\}$. Then \mathcal{F} is the set of circuits of a binary matroid.*

Proof of Proposition 2.6. Let N and N' be connected matroids and suppose that

the clutters of odd circuits of (N, Σ) and (N', Σ') are the same and are not trivial. Let M (resp., M') be the matroid constructed from (N, Σ) (resp., (N', Σ')) as in Proposition 2.8. By construction the circuits of M and M' using e are the same. Since N is connected and \mathcal{H} is not trivial, M and M' are connected. It follows from Theorem 2.7 that $M = M'$ and in particular $N = M/e = M'/e = N'$. By the same argument and Remark 2.5, $N = u(\mathcal{H})$. \square

In a binary matroid, any circuit C and cocircuit D have an even intersection. Therefore, if D is a cocircuit, the clutter of odd circuits of (M, Σ) and $(M, \Sigma \triangle D)$ are the same (see Zaslavsky [28]). Let $e \in E(M)$. The *deletion* $(M, \Sigma) \setminus e$ of (M, Σ) is defined as $(M \setminus e, \Sigma - \{e\})$. The *contraction* $(M, \Sigma)/e$ of (M, Σ) is defined as follows: If $e \notin \Sigma$, then $(M, \Sigma)/e := (M/e, \Sigma)$; if $e \in \Sigma$ and e is not a loop, then there exists a cocircuit D of M with $e \in D$ and $(M, \Sigma)/e := (M/e, \Sigma \triangle D)$. Note that if $e \in \Sigma$ is a loop of M , then \mathcal{H}/e is a trivial clutter. A *minor* of (M, Σ) is any signed matroid which can be obtained by a sequence of deletions and contractions. A minor of (M, Σ) obtained by a sequence of J_c contractions and J_d deletions is denoted $(M, \Sigma)/J_c \setminus J_d$.

Remark 2.9. Let \mathcal{H} be the clutter of odd circuits of a signed matroid (M, Σ) . If J_c does not contain an odd circuit, then $\mathcal{H}/J_c \setminus J_d$ is the clutter of odd circuits of the signed matroid $(M, \Sigma)/J_c \setminus J_d$.

Let M be a binary matroid and e an element of M . The clutter $Port(M, e)$, called a *port of M* , is defined as follows: $E(Port(M, e)) := E(M) - \{e\}$ and $Port(M, e) := \{S - \{e\} : e \in S \in \Omega(M)\}$.

PROPOSITION 2.10. *Let M be a binary matroid; then $Port(M, e)$ is a binary clutter.*

Proof. By definition $S \in Port(M, e)$ if and only if $S \cup \{e\}$ is an odd circuit of the signed matroid $(M, \{e\})$. We may assume $Port(M, e)$ is nontrivial; hence, in particular, e is not a loop of M . Therefore, there exists a cocircuit D that contains e . Thus $Port(M, e)$ is the clutter of odd circuits of the signed matroid $(M/e, D \triangle \{e\})$. Proposition 2.2 states that these odd circuits form a binary clutter. \square

PROPOSITION 2.11. *Let \mathcal{H} be a binary clutter. Then there exists a binary matroid M with element $e \in E(M) - E(\mathcal{H})$ such that $Port(M, e) = \mathcal{H}$.*

Proof. If $\emptyset \in \mathcal{H}$, define M to have element e as a loop. If $\emptyset \notin \mathcal{H}$, we can represent \mathcal{H} as the set of odd circuits of a signed matroid (N, Σ) (see Remark 2.5). Construct a binary matroid M from (N, Σ) as in Proposition 2.8. Then $Port(M, e) = \mathcal{H}$. \square

PROPOSITION 2.12 (Seymour [23]). *$Port(M, e)$ and $Port(M^*, e)$ form a blocking pair.*

Proof. Proposition 2.10 implies that $Port(M, e)$ and $Port(M^*, e)$ are both binary clutters. Consider $T \in Port(M^*, e)$. Then $T \cup \{e\}$ is a circuit of M . For all $S \in Port(M, e)$, $S \cup \{e\}$ is a circuit of M^* . Since $T \cup \{e\}$ and $S \cup \{e\}$ have an even intersection, $|S \cap T|$ is odd. Thus we proved that, for all $T \in Port(M^*, e)$, there is $T' \in b(Port(M, e))$, where $T' \subseteq T$. To complete the proof it suffices to show that, for all $T' \in b(Port(M, e))$, there is $T \in Port(M^*, e)$, where $T \subseteq T'$. Since $Port(M, e)$ is binary, for every $S \in Port(M, e)$, $|S \cap T'|$ is odd (Proposition 2.1). Thus $T' \cup \{e\}$ intersects every circuit of M using e with even parity. It follows from Theorem 2.7 that $T' \cup \{e\}$ is orthogonal to the space spanned by the circuits of M ; i.e., $T' \cup \{e\}$ is a cycle of M^* . It follows that there is a circuit of M^* of the form $T \cup \{e\}$, where $T \subseteq T'$. Hence, $T \in Port(M^*, e)$ as required. \square

3. Lifts and sources. Let N be a binary matroid. For any binary matroid M with element e such that $N = M/e$, the binary clutter $Port(M, e)$ is called a *source of N* . Note that \mathcal{H} is a source of its up matroid $u(\mathcal{H})$. For any binary matroid M

with element e such that $N = M \setminus e$, the binary clutter $Port(M, e)$ is called a *lift* of N . Note that a source or a lift can be a trivial clutter.

PROPOSITION 3.1. *Let N be a binary matroid. \mathcal{H} is a lift of N if and only if $b(\mathcal{H})$ is a source of N^* .*

Proof. Let \mathcal{H} be a lift of N ; i.e., there is a binary matroid M with $M \setminus e = N$ and $\mathcal{H} = Port(M, e)$. By Proposition 2.12, $b(\mathcal{H}) = Port(M^*, e)$. Since $M^*/e = (M \setminus e)^* = N^*$ we have that $b(\mathcal{H})$ is a source of N^* . Moreover, the implications can be reversed. \square

It is useful to relate a description of \mathcal{H} in terms of excluded *clutter minors* to a description of $u(\mathcal{H})$ in terms of excluded *matroid minors*.

THEOREM 3.2. *Let \mathcal{H} be a binary clutter such that its up matroid $u(\mathcal{H})$ is connected and let N be a connected binary matroid. Then $u(\mathcal{H})$ does not have N as a minor if and only if \mathcal{H} does not have \mathcal{H}_1 or \mathcal{H}_2^+ as a minor, where \mathcal{H}_1 is a source of N and \mathcal{H}_2 is a lift of N .*

To prove this we will need the following result (see Oxley [21, Proposition 4.3.6]).

THEOREM 3.3 (Brylawski [3], Seymour [25]). *Let M be a connected matroid and N a connected minor of M . For any $i \in E(M) - E(N)$, at least one of $M \setminus i$ or M/i is connected and has N as a minor.*

Proof of Theorem 3.2. Let $M := u(\mathcal{H})$ and let $\Sigma \in b(\mathcal{H})$. Remark 2.5 states that \mathcal{H} is the clutter of odd circuits of (M, Σ) . Suppose first that \mathcal{H} has a minor \mathcal{H}_1 that is a source of N . Remark 2.9 implies that \mathcal{H}_1 is the clutter of odd circuits of a signed minor (N', Σ') of (M, Σ) . Since N is connected, \mathcal{H}_1 is nontrivial and therefore Proposition 2.6 implies $N = N'$. In particular, N is a minor of M . Now suppose that \mathcal{H} has a minor \mathcal{H}_2^+ , where \mathcal{H}_2 is a lift of N . Let e be the element of $E(\mathcal{H}_2^+) - E(\mathcal{H}_2)$. Remark 2.9 implies that \mathcal{H}_2^+ is the clutter of odd circuits of a signed minor $(\hat{M}, \hat{\Sigma})$ of (M, Σ) . Since \mathcal{H}_2 is a lift of N there is a connected matroid \hat{M}' with element e such that $\hat{M}' \setminus e = N$ and $Port(\hat{M}', e) = \mathcal{H}_2$. Thus \mathcal{H}_2^+ is the clutter of odd circuits of $(\hat{M}', \{e\})$. Proposition 2.6 implies $\hat{M} = \hat{M}'$. Thus \hat{M}' is a minor of M and so is $N = \hat{M}' \setminus e$.

Now we prove the converse. Suppose that M has N as a minor and does not satisfy the theorem. Let \mathcal{H} be such a counterexample minimizing the cardinality of $E(\mathcal{H})$. Clearly, N is a proper minor of M as otherwise $u(\mathcal{H}) = N$; i.e., \mathcal{H} is a source of N . By Theorem 3.3, for every $i \in E(M) - E(N)$, one of $M \setminus i$ and M/i is connected and has N as a minor. Suppose M/i is connected and has N as a minor. Since i is not a loop of M , it follows from Remark 2.9 that \mathcal{H}/i is nontrivial and is a signed minor $(M/i, \Sigma')$ of (M, Σ) . Proposition 2.6 implies $M/i = u(\mathcal{H}/i)$. Therefore \mathcal{H}/i contradicts the choice of \mathcal{H} minimizing the cardinality of $E(\mathcal{H})$. Thus for every $i \in E(M) - E(N)$, $M \setminus i$ is connected and has an N minor. Suppose for some $i \in E(M) - E(N)$, $\mathcal{H} \setminus i$ is nontrivial. Then because of Remark 2.9 and Proposition 2.6 $u(\mathcal{H} \setminus i) = M \setminus i$, a contradiction to the choice of \mathcal{H} . Thus for every $i \in E(M) - E(N)$, $\mathcal{H} \setminus i$ is trivial, or, equivalently, all odd circuits of (M, Σ) use i . As $M = u(\mathcal{H})$, even circuits of M do not use i . We claim that $E(M) - E(N) = \{i\}$. Suppose not and let $j \neq i$ be an element of $E(M) - E(N)$. The set of circuits of (M, Σ) using j is exactly the set of odd circuits. It follows that the elements i, j must be in series in M . Thus $M \setminus i$ is not connected, a contradiction. Therefore $E(M) - E(N) = \{i\}$ and $M \setminus i = N$. As the circuits of (M, Σ) using i are exactly the odd circuits of (M, Σ) , it follows that column i of $A(\mathcal{H})$ consists of all 1's. Thus $\mathcal{H} = \mathcal{H}_2^+$, where $\mathcal{H}_2 = Port(M, i)$; i.e., \mathcal{H}_2 is a lift of N . \square

Next we define the binary matroids F_7, F_7^* , and R_{10} . For any binary matroid N ,

let B_N be a 0, 1 matrix whose rows span the circuit space of N . (Equivalently, B_N is a representation of the dual matroid N^* .) Square identity matrices are denoted I . Observe that $R_{10}^* = R_{10}$.

$$B_{F_7} = \left[I \mid \begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \right], \quad B_{F_7^*} = \left[I \mid \begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{array} \right],$$

$$B_{R_{10}} = \left[I \mid \begin{array}{ccccc} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{array} \right].$$

Given a binary matroid N , let M be a binary matroid with element e such that $N = M/e$. The circuit space of M is spanned by the rows of a matrix of the form $[B_N|x]$, where x is a 0, 1 column vector indexed by e . Assuming M is connected, we have (up to isomorphism) the following possible columns x for each of the three aforementioned matroids N :

- (1) F_7^* :
 $x_a = (1, 1, 1)^T$.
- (2) F_7 :
 $x_a = (0, 1, 1, 1)^T$, $x_b = (1, 1, 1, 0)^T$, and $x_c = (1, 1, 1, 1)^T$.
- (3) R_{10} :
 $x_a = (1, 0, 1, 0, 0)^T$, $x_b = (1, 0, 1, 0, 1)^T$, $x_c = (1, 0, 1, 1, 0)^T$,
 $x_d = (1, 0, 1, 1, 1)^T$, $x_e = (1, 1, 0, 0, 0)^T$, $x_f = (1, 1, 1, 1, 1)^T$.

Note that (1), (2) are easy and (3) can be found in [24, p. 357]. The rows of the matrix $[B_{F_7}|x_b]$ (resp., $[B_{F_7}|x_c]$) span the circuit space of a matroid known as $AG(3, 2)$ (resp., S_8). If $[B_N|x]$ is a matrix whose rows span the circuits of M , then by definition of sources $Port(M, e)$ is a source of N . Thus, we have the following remark.

Remark 3.4.

- F_7^* has a unique source, namely Q_6^+ .
- F_7 has three sources: $b(Q_6)^+$ (when $x = x_a$), \mathcal{L}_{F_7} (when $x = x_b$), and $b(Q_7)$ (when $x = x_c$).
- R_{10} has six sources including $b(\mathcal{O}_{K_5})$ (when $x = x_f$).

Luetolf and Margot [16] have enumerated all mni clutters with at most 10 elements (and many more). Using Remark 3.4, we can then readily check the following.

PROPOSITION 3.5. *Let \mathcal{H} be the clutter of odd circuits of a signed matroid (M, Σ) .*

- *If $M = R_{10}$, then either $\mathcal{H} = b(\mathcal{O}_{K_5})$ or \mathcal{H} is ideal.*
- *If $M = F_7$, then either $\mathcal{H} = \mathcal{L}_{F_7}$ or \mathcal{H} is ideal.*
- *If $M = F_7^*$, then \mathcal{H} is ideal.*

4. Multicommodity flows. In this section, we show that a binary clutter \mathcal{H} is ideal exactly when certain multicommodity flows exist in the matroid $u(\mathcal{H})$. This equivalence will be used in sections 6 and 7 to show that mni binary clutters do not have small separations. Given a set S , a function $p : S \rightarrow \mathbb{Q}_+$, and $T \subseteq S$, we write $p(T)$ for $\sum_{i \in T} p(i)$. Consider a signed matroid (M, F) . The set of circuits of M that have exactly one element in common with F is denoted Ω_F . Let $p : E(M) \rightarrow \mathbb{Q}_+$ be a cost function on the elements of M . Seymour [26] considers the following two statements about the triple (M, F, p) .

For any cocircuit D of M

$$(4.1) \quad p(D \cap F) \leq p(D - F).$$

There exists a function $\eta : \Omega_F \rightarrow \mathbb{Q}_+$ such that

$$(4.2) \quad \sum_{C: e \in C \in \Omega_F} \eta(C) \begin{cases} \geq p(e) & \text{if } e \in F, \\ \leq p(e) & \text{if } e \in E - F. \end{cases}$$

We say that *the cut condition holds* if inequality (4.1) holds for all cocircuits D . We say that M is *F -flowing with costs p* if statement (4.2) holds; the corresponding solution is an *F -flow satisfying costs p* . M is *F -flowing* [26] if, for every p for which the cut condition holds, M is F -flowing with costs p . Elements in F (resp., $E - F$) are called *demand* (resp., *capacity*) elements. It is helpful to illustrate the aforementioned definitions in the case where M is a graphic matroid [9]. For a demand edge f , $p(f)$ is the amount of flow required between its endpoints. For a capacity edge e , $p(e)$ is the maximum amount of flow that can be carried by e . Then M is F -flowing with costs p when a multicommodity flow meeting all demands and satisfying all capacity constraints exists. The cut condition requires that for every cut the demand across the cut does not exceed its capacity. When F consists of a single edge f and when M is graphic, then M is f -flowing [7].

The cut condition states that $p(D \cap F) \leq p(D - F) = p(D) - p(D \cap F)$. Adding $p(F) - p(D \cap F)$ to both sides, we obtain $p(F) \leq p(D) - p(D \cap F) + p(F) - p(D \cap F) = p(D \triangle F)$. Hence, we have the following remark.

Remark 4.1. The cut condition holds if and only if $p(F) \leq p(D \triangle F)$ for all cocircuits D .

Let \mathcal{H} be the clutter of odd circuits of (M, F) . We define

$$(a) \quad \tau(\mathcal{H}, p) = \min \left\{ \sum_{e \in E(M)} p(e)x_e : \sum_{e \in S} x_e \geq 1 \ \forall S \in \mathcal{H}, x_e \in \{0, 1\} \ \forall e \in E(M) \right\},$$

$$(b) \quad \nu^*(\mathcal{H}, p) = \max \left\{ \sum_{C \in \mathcal{H}} y_C : \sum_{C: e \in C \in \mathcal{H}} y_C \leq p(e) \ \forall e \in E(M), y_C \geq 0 \ \forall C \in \mathcal{H} \right\}.$$

By linear programming duality we have $\tau(\mathcal{H}, p) \geq \nu^*(\mathcal{H}, p)$. When $p(e) = 1$ for all $e \in E(M)$, then we write $\tau(\mathcal{H})$ for $\tau(\mathcal{H}, p)$ and $\nu^*(\mathcal{H})$ for $\nu^*(\mathcal{H}, p)$.

PROPOSITION 4.2. *Let \mathcal{H} be the clutter of odd circuits of a signed matroid (M, F) and let $p : E(M) \rightarrow \mathbb{Q}_+$.*

- (i) $\tau(\mathcal{H}, p) = p(F)$ if and only if the cut condition holds.
- (ii) $\nu^*(\mathcal{H}, p) = p(F)$ if and only if M is F -flowing with costs p .
- (iii) If $\eta(C) > 0$ for a solution to 4.2, then $C \in \Omega_F$ for all $F \in b(\mathcal{H})$ with $p(F) = \tau(\mathcal{H}, p)$.

Proof. We say that a set $X \subseteq E(M)$ is a (feasible) solution for (a) if its characteristic vector is. Consider (i). Suppose $\tau(\mathcal{H}, p) = p(F)$. We can assume that F is an inclusion-wise minimal solution of (a) and thus $F \in b(\mathcal{H})$. Let D be any cocircuit of M and consider any $S \in \mathcal{H}$. Since S is a circuit of M , $|D \cap S|$ is even and since \mathcal{H} is binary, $|F \cap S|$ is odd. Thus $|(D \triangle F) \cap S|$ is odd. It follows that $D \triangle F$ is a transversal of \mathcal{H} . Therefore, $D \triangle F$ is a feasible solution to (a) and we have $p(F) \leq p(D \triangle F)$. Hence, by Remark 4.1, the cut condition holds. Conversely, assume the cut condition holds and consider any set X that is feasible for (a). We need to show $p(F) \leq p(X)$.

We can assume that X is inclusion-wise minimal, i.e., that $X \in b(\mathcal{H})$. Observe that F and X intersect circuits of M with the same parity. Thus $D := F \triangle X$ is a cocycle of M . Since the cut condition holds, by Remark 4.1, $p(F) \leq p(D \triangle F) = p(X)$.

Consider (ii). Suppose $\nu^*(\mathcal{H}, p) = p(F)$. Since $\nu^*(\mathcal{H}, p) \leq \tau(\mathcal{H}, p) \leq p(F)$, it follows from linear programming duality that F is an optimal solution to (a). Let y be an optimal solution to (b). Complementary slackness states that if $|F \cap C| > 1$, then the corresponding dual variable $y_C = 0$. Thus $\sum_{C: e \in C \in \mathcal{H}} y_C = \sum_{C: e \in C \in \Omega_F} y_C$ for all $e \in E(M)$. Complementary slackness states that if $e \in F$, then $\sum_{C: e \in C \in \Omega_F} y_C = p(e)$. Hence, choosing $\eta(C) = y_C$ for every $C \in \Omega_F$ satisfies (4.2). Conversely, suppose η is a solution to (4.2). For each $e \in F$ such that $\sum_{C: e \in C \in \Omega_F} \eta_C > p(e)$, reduce the values η_C on the left-hand side until equality holds. Since C contains no element of F other than e , we can get equality for every $e \in F$. So we may assume $\sum_{C: e \in C \in \Omega_F} \eta(C) \leq p(e)$ for all $e \in E(M)$. Set $y_C = 0$ if $C \notin \Omega_F$ and $y_C = \eta(C)$ if $C \in \Omega_F$. Now y is a feasible solution to (b) and F, y satisfy all complementary slackness conditions. Thus F and y must be a pair of optimal solutions to (a) and (b), respectively.

Finally, consider (iii). From (ii) we know there is an optimal solution y to (b) with $y_C > 0$. By complementary slackness, it follows that $|F \cap C| = 1$ for all F that are optimal solutions to (a). \square

The last proposition implies in particular that, if M is F -flowing with costs p , then the cut condition is satisfied. We say that a cocircuit D is *tight* if the cut condition (4.1) holds with equality, or, equivalently (Remark 4.1), if $p(F) = p(D \triangle F)$.

PROPOSITION 4.3. *Suppose M is F -flowing with costs p and let D be a tight cocircuit. If C is a circuit with $\eta(C) > 0$, then $C \cap D = \emptyset$ or $C \cap D = \{e, f\}$, where $e \in E(M) - F$ and $f \in F$.*

Proof. We may assume $C \cap D \neq \emptyset$. As $C \in \Omega_F$, it follows that $C \cap F = \{f\}$. Moreover, $C \cap D \neq \{f\}$, since M is binary. To complete the proof, it suffices to show that there is no pair of elements $e, e' \in (C \cap D) - F$. Suppose for a contradiction that we have such a pair and let $F' = D \triangle F$. As D is tight, $p(F) = p(D \triangle F) = p(F')$. It follows from Proposition 4.2(iii) that $C \in \Omega_{F'}$. However, $e, e' \in F'$, a contradiction. \square

COROLLARY 4.4. *Let \mathcal{H} be the clutter of odd circuits of a signed matroid (M, F) . (i) If \mathcal{H} is ideal, then M is F -flowing with costs p for all $p : E(M) \rightarrow \mathbb{Q}_+$, where (M, F, p) satisfies the cut condition. (ii) If \mathcal{H} is nonideal, then M is not F' -flowing with costs p for some $p : E(M) \rightarrow \mathbb{Q}_+$ and some $F' \in b(\mathcal{H})$ that minimizes $p(F')$.*

Proof. Consider (i). Proposition 4.2 states that $\tau(\mathcal{H}, p) = p(F)$. Because \mathcal{H} is ideal, $\tau(\mathcal{H}, p) = \nu^*(\mathcal{H}, p)$, i.e., $p(F) = \nu^*(\mathcal{H}, p)$. This implies by Proposition 4.2(ii) that M is F -flowing with costs p . Consider (ii). If \mathcal{H} is nonideal, then for some $p : E(M) \rightarrow \mathbb{Z}_+$, $\tau(\mathcal{H}, p) > \nu^*(\mathcal{H}, p)$ [5]. Let F' be an optimal solution to (a). Then $p(F') = \tau(\mathcal{H}, p)$ and $F' \in b(\mathcal{H})$. Proposition 4.2(ii) states that M is not F' -flowing with costs p . \square

We leave the next result as an easy exercise.

COROLLARY 4.5. *A binary clutter \mathcal{H} is ideal if and only if $u(\mathcal{H})$ is F -flowing for every $F \in b(\mathcal{H})$.*

Consider the case where $\mathcal{H} = \mathcal{O}_{K_5}$. Let F be a set of edges of K_5 such that $E(K_5) - F$ induces a $K_{2,3}$. Then $F \in b(\mathcal{H})$ and $u(\mathcal{H})$ (the graphic matroid of K_5) is not F -flowing.

5. Connectivity, preliminaries. Let E_1, E_2 be a partition of the elements E of a matroid M and let $r : 2^{|E|} \rightarrow \mathbb{Z}_+$ be the rank function. M is said to have a k -separation E_1, E_2 if $r(E_1) + r(E_2) - r(E) \leq k - 1$ and $|E_1|, |E_2| \geq k$. If $|E_1|, |E_2| > k$,

then the separation is said to be *strict*. A matroid M has a k -separation E_1, E_2 if and only if its dual M^* does (Oxley [21, 4.2.7]). A matroid is k -connected if it has no $(k - 1)$ -separation and is *internally k -connected* if it has no strict $(k - 1)$ -separation. A 2-connected matroid is simply said to be *connected*. We now follow Seymour [24] when presenting k -sums. Let M_1, M_2 be binary matroids whose element sets $E(M_1), E(M_2)$ may intersect. We define $M_1 \triangle M_2$ to be the binary matroid on $E(M_1) \triangle E(M_2)$, where the cycles are all the subsets of $E(M_1) \triangle E(M_2)$ of the form $C_1 \triangle C_2$, where C_i is a cycle of $M_i, i = 1, 2$. The following special cases will be of interest to us.

DEFINITION 5.1.

1. $E(M_1) \cap E(M_2) = \emptyset$. Then $M_1 \triangle M_2$ is the 1-sum of M_1, M_2 .
2. $E(M_1) \cap E(M_2) = \{f\}$ and f is not a loop of M_1 or M_2 . Then $M_1 \triangle M_2$ is the 2-sum of M_1, M_2 .
3. $|E(M_1) \cap E(M_2)| = 3$ and $E(M_1) \cap E(M_2)$ is a circuit of both M_1 and M_2 . Then $M_1 \triangle M_2$ is the 3-sum of M_1, M_2 .

We denote the k -sum of M_1 and M_2 as $M_1 \otimes_k M_2$. The elements in $E(M_1) \cap E(M_2)$ are called the *markers* of $M_i (i = 1, 2)$. As an example, for $k = 1, 2, 3$, the k -sum of two graphic matroids corresponds to taking two graphs, choosing a k -clique from each, identifying the vertices in the clique pairwise and deleting the edges in the clique. The markers are the edges in the clique. We have the following connection between k -separations and k -sums.

THEOREM 5.2 (Seymour [24]). *Let M be a k -connected binary matroid and $k \in \{1, 2, 3\}$. Then M has a k -separation if and only if it can be expressed as $M_1 \otimes_k M_2$. Moreover, M_1 (resp., M_2) is a minor of M obtained by contracting and deleting elements in $E(M_2) - E(M_1)$ (resp., $E(M_1) - E(M_2)$).*

We say that a binary clutter \mathcal{H} has a (strict) k -separation if $u(\mathcal{H})$ does.

Remark 5.3. \mathcal{H} has a 1-separation if and only if $A(\mathcal{H})$ is a block diagonal matrix. Moreover, \mathcal{H} is ideal if and only if the minors corresponding to each of the blocks are ideal.

Recall (Proposition 2.11) that every binary clutter \mathcal{H} can be expressed as $Port(M, e)$ for some binary matroid M with element e . Therefore we could define the connectivity of \mathcal{H} to be the connectivity of the associated matroid M . The two notions of connectivity are not equivalent as the clutter \mathcal{L}_{F_7} illustrates. The matroid $AG(3, 2)$ has a strict 3-separation while F_7 does not, but $Port(AG(3, 2), t) = \mathcal{L}_{F_7}$ and \mathcal{L}_{F_7} is the clutter of odd circuits of the signed matroid $(F_7, E(F_7))$.

Chopra [4] gives composition operations for matroid ports and sufficient conditions for maintaining idealness. This generalizes earlier results of Bixby [1]. Other compositions for ideal (but not necessarily binary clutters) can be found in [19, 17, 18]. Novick and Sebö [20] give an outline on how to show that mni binary clutters do not have 2-separations; the argument is similar to that used by Seymour [26, 7.1] to show that k -cycling matroids are closed under 2-sums. We will follow the same strategy (see section 6). Proving that mni binary clutters do not have 3-separations is more complicated and requires a different approach (see section 7). In closing observe that none of $\mathcal{L}_{F_7}, \mathcal{O}_{K_5}$, and $b(\mathcal{O}_{K_5})$ have strict 4-separations. Therefore, if Seymour’s conjecture holds, then mni binary clutters are internally 5-connected.

6. 2-separations. Let (M, F) be a signed matroid with a 2-separation E_1, E_2 , i.e., $M = M_1 \otimes_2 M_2$ and $E_1 = E(M_1) - E(M_2), E_2 = E(M_2) - E(M_1)$. We say that (M_i, F_i) (for $i = 1, 2$) is a *part* of (M, F) if it is a signed minor of (M, F) . It is not hard to see that at most two choices of F_i can give distinct signed matroids (M_i, F_i) . Therefore (M, F) can have at most four distinct parts. In light of Remark 2.5 we can

identify binary clutters with signed matroids. The main result of this section is the following.

PROPOSITION 6.1. *A binary clutter with a 2-separation is ideal if and only if all its parts are ideal.*

To prove this, we shall need the following results.

PROPOSITION 6.2 (Seymour [24]). *If $M = M_1 \otimes_2 M_2$, then M is connected if and only if M_1 and M_2 are connected.*

PROPOSITION 6.3 (Seymour [24]). *Let M be a binary matroid with a 2-separation E_1, E_2 and let C_1, C_2 be two circuits of M . If $C_1 \cap E_i \subseteq C_2 \cap E_i$, then $C_1 \cap E_i = C_2 \cap E_i$ (for $i = 1, 2$).*

PROPOSITION 6.4 (Seymour [24]). *Let $M = M_1 \otimes_2 M_2$. Then choose any circuit C of M such that $C \cap E_1 \neq \emptyset$ and $C \cap E_2 \neq \emptyset$. Let $i, j = 1, 2$ and $i \neq j$. For any $f \in C \cap E_j$, $M_i = M \setminus (E_j - C) / (E_j \cap C - \{f\})$.*

Proof of Proposition 6.1. Let \mathcal{H} be a binary clutter with a 2-separation, $M = u(\mathcal{H})$ and $F \in b(\mathcal{H})$. Assume without loss of generality that M is connected. Remark 2.5 states that \mathcal{H} is the clutter of odd circuits of (M, F) . If \mathcal{H} is ideal, then so are all its parts by Remark 2.9. Conversely, suppose all parts of (M, F) are ideal. Consider any $p : E(M) \rightarrow \mathbb{Z}_+$ and assume $F \in b(\mathcal{H})$ minimizes $p(F)$. Because of Corollary 4.4(ii), it suffices to show that M is F -flowing with costs p . Observe that the cut condition is satisfied because of Proposition 4.2(i).

Since M has a 2-separation, it can be expressed as $M_1 \otimes_2 M_2$. Throughout this proof, i, j will always denote arbitrary distinct elements of $\{1, 2\}$. Define $F_i = F \cap E_i$ and let f_i be the marker of M_i . Since f_i is not a loop, there is a cocircuit D_i of M_i using f_i . Let α_i denote the smallest value of

$$(*) \quad p[D_i - (F_i \cup \{f_i\})] - p(D_i \cap F_i),$$

where D_i is any cocircuit of M_i using f_i . In what follows, we let D_i denote some cocircuit where the minimum is attained. Expression (*) gives the difference between the sum of the capacity elements and the sum of the demand elements in D_i , excluding the marker f_i . Thus $\alpha_1 + \alpha_2 = p([D_1 \Delta D_2] - F) - p([D_1 \Delta D_2] \cap F)$. Since $D_1 \Delta D_2$ is a cycle of M and the cut condition is satisfied, we must have

$$\alpha_1 + \alpha_2 \geq 0.$$

Claim 1. If $\alpha_i > 0$, then there is an even circuit of (M_i, F_i) that uses marker f_i .

Proof. Suppose for a contradiction that all circuits C of M_i that use f_i satisfy $|C \cap F_i|$ odd. Then $D = F_i \cup \{f_i\}$ intersects all these circuits with even parity. By hypothesis, M is connected and, because of Proposition 6.2, so is M_i . We know from Theorem 2.7 that all circuits that do not use the marker f_i are the symmetric difference of two circuits that do use f_i . It follows that D intersects all circuits of M_i with even parity. Thus D is a cocycle of M_i . However, expression (*) is nonpositive for cocycle D . D can be partitioned into cocircuits. Because the cut condition holds, expression (*) is nonpositive for the cocircuit that uses f_i , a contradiction as $\alpha_i > 0$. \square

Claim 2. If $\alpha_i < 0$, then there is an odd circuit of (M_i, F_i) that uses marker f_i .

Proof. Suppose, for a contradiction, that all circuits C of M_i that use f_i satisfy $|C \cap F_i|$ even. By the same argument as in Claim 1, we know that in fact so do all circuits of M_i . This implies that F and F_j intersect each circuit of M with the same parity. As F is inclusion-wise minimal ($F \in b(\mathcal{H})$) we must have $F = F_j$, i.e., $F_i = \emptyset$. However, this implies that expression (*) is nonnegative, a contradiction. \square

Claim 3. If $\alpha_j < 0$ (resp., $\alpha_j > 0$), then $(M_i, F_i \cup \{f_i\})$ (resp., (M_i, F_i)) is a part of (M, F) .

Proof. From Claim 2 (resp., Claim 1) there is an odd (resp., even) circuit C using f_j of (M_j, F_j) . Proposition 6.3 implies that elements $C \cap E$ are in series in $M \setminus (E_j - C)$. Proposition 6.4 implies that M_i is obtained from $M \setminus (E_j - C)$ by replacing series elements of $C \cap E_j$ by a unique element f_j . The required signed minor is $(M, F) \setminus (E_j - C) / (C - \{f\})$, where f is any element of $C \cap E_j$. \square

Because $\alpha_1 + \alpha_2 \geq 0$, it suffices to consider the following cases.

Case 1. $\alpha_1 \geq 0, \alpha_2 \geq 0$.

We know from Proposition 6.4 that M_i is a minor of M (where no loop is contracted), say $M \setminus J_d / J_c$. For $i = 1, 2$, let (M_i, \hat{F}_i) be the signed minor $(M, F) \setminus J_d / J_c$. Since (M_i, \hat{F}_i) is a part of (M, F) , it is ideal. Therefore in particular $(M_i, \hat{F}_i) \setminus f_i = (M_i \setminus f_i, F_i)$ are ideal. Let $p^i : E(M_i) - f_i \rightarrow \mathbb{Z}_+$ be defined as follows: $p^i(e) = p(e)$ if $e \in E_i$. Let D be a cocircuit of $M_i \setminus f_i$. The inequality $p(D \cap F_i) \leq p(D - F_i)$ follows from $\alpha_i \geq 0$ when $D \cup f_i$ is a cocircuit of M_i , and it follows from the fact that the cut condition holds for (M, F) when D is a cocircuit of M_i . Therefore the cut condition holds for $(M_i \setminus f_i, F_i)$. It follows from Corollary 4.4(i) that each of these signed matroids has an F_i -flow satisfying costs p^i . Let $\eta_i = \Omega_{F_i} \rightarrow \mathbb{Q}_+$ be the corresponding function satisfying (4.2). By scaling p , we may assume $\eta_i(C) \in \mathbb{Z}_+$ for each circuit in Ω_{F_i} . Let \mathcal{L}_i be the multiset where each circuit C in Ω_{F_i} appears $\eta_i(C)$ times. Define \mathcal{L}_j similarly. The union (with repetition) of all circuits in \mathcal{L}_i and \mathcal{L}_j correspond to an F -flow of M satisfying costs p .

Case 2. $\alpha_i < 0, \alpha_j > 0$.

Because of Claim 3, there are parts (M_i, F_i) and $(M_j, F_j \cup \{f_j\})$ of (M, F) . Let $p^i : E(M_i) \rightarrow \mathbb{Z}_+$ be defined as follows: $p^i(f_i) = \alpha_j$ and $p^i(e) = p(e)$ for $e \in E_i$. Let $p^j : E(M_j) \rightarrow \mathbb{Z}_+$ be defined as follows: $p^j(f_j) = -\alpha_i$ and $p^j(e) = p(e)$ for $e \in E_j$. Since we can scale p , we can assume that the F_i -flow of M_i satisfying costs p^i is a multiset \mathcal{L}^i of circuits and that the $F_j \cup \{f_j\}$ -flow of M_j satisfying costs p^j is a multiset \mathcal{L}^j . For $l = 1, 2$, \mathcal{L}^l can be partitioned into $\mathcal{L}_0^l := \{C \in \mathcal{L}^l : f_l \notin C\}$ and $\mathcal{L}_1^l := \{C \in \mathcal{L}^l : f_l \in C\}$. Since f_j is a demand element for the flow \mathcal{L}^j , $|\mathcal{L}_1^j| = p^j(f_j) = -\alpha_i$. Since f_i is a capacity element for the flow \mathcal{L}^i , $|\mathcal{L}_1^i| \leq p^i(f_i) = \alpha_j$. Because $\alpha_1 + \alpha_2 \geq 0$, $|\mathcal{L}_1^j| \leq |\mathcal{L}_1^i|$. Let us define a collection of circuits of M as follows: include all circuits of $\mathcal{L}_0^i \cup \mathcal{L}_0^j$. Pair each circuit $C_j \in \mathcal{L}_1^j$ with a different circuit $C_i \in \mathcal{L}_1^i$ and add to the collection the circuit included in $C_i \triangle C_j$ that contains the element of F . The resulting collection corresponds to an F -flow of M satisfying costs p . \square

7. 3-separations. The main result of this section is the following,

PROPOSITION 7.1. *A mni binary clutter \mathcal{H} has no strict 3-separation.*

The proof follows from two lemmas, stated next and proved in sections 7.1 and 7.2, respectively.

LEMMA 7.2. *Let \mathcal{H} be a mni binary clutter with a strict 3-separation E_1, E_2 . There exists a set $F \in b(\mathcal{H})$ of minimum cardinality such that $F \subseteq E_1$ or $F \subseteq E_2$.*

Let (M, F) be a signed matroid with a strict 3-separation E_1, E_2 , i.e., $M = M_1 \otimes_3 M_2$ and $E_1 = E(M_1) - E(M_2), E_2 = E(M_2) - E(M_1)$. Let $C_0 = E(M_1) \cap E(M_2)$ be the triangle common to both M_1 and M_2 . Let \bar{M}_i (with $i = 1, 2$) be obtained by deleting from M_i a (possibly empty) set of elements of C_0 . We call (\bar{M}_i, F_i) a part of (M, F) if it is a signed minor of (M, F) .

LEMMA 7.3. *Let (M, F) be a connected signed matroid with a strict 3-separation E_1, E_2 and suppose $F \subseteq E_1$. Then M is F -flowing with costs p if the cut condition is satisfied and all parts of (M, F) are ideal.*

Proof of Proposition 7.1. Suppose \mathcal{H} is a mni binary clutter that is connected with a strict 3-separation. Remark 2.5 states that \mathcal{H} is the clutter of odd circuits of a signed matroid (M, F) . Consider $p : E(M) \rightarrow \mathbb{Z}_+$ defined by $p(e) = 1$ for all $e \in E(M)$. We know (see Remark 7.5) that $\tau(\mathcal{H}, p) > \nu^*(\mathcal{H}, p)$. From Lemma 7.2 and Remark 2.5, we may assume $F \subseteq E_1$ and $p(F) = \tau(\mathcal{H}, p)$. It follows from Proposition 4.2(i) that the cut condition holds. Since the separation of (M, F) is strict, all parts of (M, F) are proper minors, and hence ideal. It follows therefore from Lemma 7.3 that M is F -flowing with costs p . Hence, because of Proposition 4.2(ii), $\nu^*(\mathcal{H}, p) = p(F)$, a contradiction. \square

7.1. Separations and blocks. In this section, we shall prove Lemma 7.2. However, first let us review some results on mni clutters. For every clutter \mathcal{H} , we can associate a 0, 1 matrix $A(\mathcal{H})$. Hence we shall talk about mni 0, 1 matrices, blockers of 0, 1 matrices, and binary 0, 1 matrices (when the associated clutter is binary). The next result on mni 0, 1 matrices is due to Lehman [15] (see also Padberg [22], Seymour [27]). We state it here in the binary case.

THEOREM 7.4. *Let A be a mni binary 0, 1 matrix with n columns. Then $B = b(A)$ is mni binary as well, and the matrix A (resp., B) has a square, nonsingular row submatrix \bar{A} (resp., \bar{B}) with r (resp., s) nonzero entries in every row and column, $rs > n$. Rows of A (resp., B) not in \bar{A} (resp., \bar{B}) have at least $r + 1$ (resp., $s + 1$) nonzero entries. Moreover, $\bar{A}\bar{B}^T = J + (rs - n)I$, where J denotes an $n \times n$ matrix filled with ones.*

It follows that $(\frac{1}{r}, \dots, \frac{1}{r})$ is a fractional extreme point of the polyhedron $\{x \in \mathbb{R}_+^n : Ax \geq \mathbf{1}\}$. Hence, we have the following remark.

Remark 7.5. If \mathcal{H} is a mni binary clutter, then $\tau(\mathcal{H}) > \nu^*(\mathcal{H})$.

The submatrix \bar{A} is called the *core* of A . Given a mni clutter \mathcal{H} with $A = A(\mathcal{H})$, we define the core of \mathcal{H} to be the clutter $\bar{\mathcal{H}}$ for which $A(\bar{\mathcal{H}}) = \bar{A}$. Let \mathcal{H} and $\mathcal{G} = b(\mathcal{H})$ be binary and mni. Since \mathcal{H}, \mathcal{G} are binary, for all $S \in \bar{\mathcal{H}}$ and $T \in \bar{\mathcal{G}}$, we have $|S \cap T|$ odd. As $\bar{A}\bar{B}^T = J + (rs - n)I$, for every $S \in \bar{\mathcal{H}}$, there is exactly one set $T \in \bar{\mathcal{G}}$ called the *mate* of S such that $|S \cap T| = 1 + (rs - n)$. Note that if A is binary, then $rs - n + 1 \geq 3$.

PROPOSITION 7.6. *Let A be a mni binary matrix. Then no column of \bar{A} is in the union of two other columns.*

Proof. Bridges and Ryser [2] proved that square 0, 1 matrices \bar{A}, \bar{B} that satisfy $\bar{A}\bar{B}^T = J + (rs - n)I$ commute, i.e., $\bar{A}^T\bar{B} = J + (rs - n)I$. Thus $col(\bar{A}, i)col(\bar{B}, i) = rs - n + 1 \geq 3$ for every $i \in \{1, \dots, n\}$. Hence there is no $j, k \in \{1, \dots, n\} - \{i\}$ such that $col(\bar{A}, j) \cup col(\bar{A}, k) \supseteq col(\bar{A}, i)$, for otherwise $col(\bar{A}, j)col(\bar{B}, i) > 1$ or $col(\bar{A}, k)col(\bar{B}, i) > 1$, contradicting the equation $\bar{A}^T\bar{B} = J + (rs - n)I$. \square

PROPOSITION 7.7 (Guenin [10]). *Let \mathcal{H} be a mni binary clutter and $e \in E(\mathcal{H})$. There exist $S_1, S_2, S_3 \in \bar{\mathcal{H}}$ such that $S_1 \cap S_2 = S_2 \cap S_3 = S_1 \cap S_3 = \{e\}$.*

PROPOSITION 7.8 (Guenin [10]). *Let \mathcal{H} be a mni binary clutter and $S_1, S_2 \in \bar{\mathcal{H}}$. If $S \subseteq S_1 \cup S_2$ and $S \in \mathcal{H}$, then either $S = S_1$ or $S = S_2$.*

PROPOSITION 7.9. *Let \mathcal{H} be a mni binary clutter and let $S, S' \in \bar{\mathcal{H}}$. Then $|S - S'| \geq 2$.*

Proof. Let T be the mate of S . Then $|T \cap S| \geq 3$ and $|T \cap S'| = 1$. \square

PROPOSITION 7.10 (Luetolf and Margot [16]). *Let \mathcal{H} be a mni binary clutter. Then $\tau(\mathcal{H}) = \tau(\bar{\mathcal{H}})$. Furthermore, if T is a transversal of $\bar{\mathcal{H}}$ and $|T| = \tau(\bar{\mathcal{H}})$, then T is a transversal of \mathcal{H} .*

We shall also need the following proposition.

PROPOSITION 7.11 (Seymour [24]). *Let M be a binary matroid with 3-separation E_1, E_2 . Then there exist circuits C_1, C_2 such that every circuit of M can be expressed as the symmetric difference of a subset of circuits in $\{C \in \Omega(M) : C \subseteq E_1 \text{ or } C \subseteq E_2\} \cup \{C_1, C_2\}$.*

Throughout this section, we shall consider a signed matroid (M, F) with a 3-separation E_1, E_2 , and C_1, C_2 will denote the corresponding circuits of Proposition 7.11. Let \mathcal{H} be the clutter of odd circuits of (M, F) . We shall partition $b(\mathcal{H})$ into sets B_1, B_2, B_3, B_4 as follows:

$$\begin{aligned} B_1 &= \{S \in b(\mathcal{H}) : |S \cap C_1 \cap E_1| \text{ even, } |S \cap C_2 \cap E_1| \text{ even}\}, \\ B_2 &= \{S \in b(\mathcal{H}) : |S \cap C_1 \cap E_1| \text{ even, } |S \cap C_2 \cap E_1| \text{ odd}\}, \\ B_3 &= \{S \in b(\mathcal{H}) : |S \cap C_1 \cap E_1| \text{ odd, } |S \cap C_2 \cap E_1| \text{ even}\}, \\ B_4 &= \{S \in b(\mathcal{H}) : |S \cap C_1 \cap E_1| \text{ odd, } |S \cap C_2 \cap E_1| \text{ odd}\}. \end{aligned}$$

PROPOSITION 7.12. *If $S_1, S_2 \in B_i$ where $i \in \{1, \dots, 4\}$, then $(S_1 \cap E_1) \cup (S_2 \cap E_2)$ contains a set of $b(\mathcal{H})$.*

Proof. Let $S' := (S_1 \cap E_1) \cup (S_2 \cap E_2)$. Note that since $S_1, S_2 \in b(\mathcal{H})$ for all circuits C of M , $|S_1 \cap C|$ and $|S_2 \cap C|$ have the same parity. This implies that if C is a circuit where $C \subseteq E_k$ ($k \in \{1, 2\}$), then C intersects S' and S_1 with the same parity. It also implies, together with the definition of B_i , that S' intersects C_k ($k \in \{1, 2\}$) with the same parity as S_1 . It follows from Proposition 7.11 that S' and S_1 intersect all circuits of M with the same parity. \square

Proof of Lemma 7.2. Let \mathcal{G} denote the blocker of \mathcal{H} and let B_1, B_2, B_3, B_4 be the sets partitioning \mathcal{G} . We will denote by $\bar{\mathcal{G}}$ the core of \mathcal{G} . It follows that $\bar{\mathcal{G}}$ can be partitioned into sets \bar{B}_i with $i \in \{1, \dots, 4\}$ and $\bar{B}_i \subseteq B_i$. Assume for a contradiction that for all $S \in \bar{\mathcal{G}}$, $S \cap E_1 \neq \emptyset \neq S \cap E_2$. We will say that a set \bar{B}_i with $i \in \{1, \dots, 4\}$ forms an E_1 -block if, for all pairs of sets $S, S' \in \bar{B}_i$, we have $S \cap E_1 = S' \cap E_1 \neq \emptyset$. Similarly we define E_2 -blocks.

Claim 1. For $i \in \{1, \dots, 4\}$, each nonempty \bar{B}_i is either an E_1 - or an E_2 -block.

Proof. Consider S_1, S_2 in \bar{B}_i . Proposition 7.12 states that $(S_1 \cap E_1) \cup (S_2 \cap E_2)$ contains a set $S' \in \mathcal{G}$. Proposition 7.8 implies that $S' = S_1$ or $S' = S_2$. If $S' = S_1$, then $(S_1 \cap E_2) = (S_2 \cap E_2)$. If $S' = S_2$, then $(S_1 \cap E_1) = (S_2 \cap E_1)$. Moreover, by hypothesis neither $S' \cap E_1$ nor $S' \cap E_2$ is empty. Since S, S' were chosen arbitrarily, the result follows. \square

For any nonempty \bar{B}_i , consider any $S \in \bar{B}_i$. We define $E(\bar{B}_i)$ to be equal to $S \cap E_1$ if \bar{B}_i is an E_1 -block and to $S \cap E_2$ if \bar{B}_i is an E_2 -block. Let r (resp., s) be the cardinality of the members of $\bar{\mathcal{H}}$ (resp., $\bar{\mathcal{G}}$) and $n = |E(\bar{\mathcal{H}})|$. As \mathcal{H} is binary, $r \geq 3$ and $s \geq 3$.

Claim 2. Let $U \subseteq E(\bar{\mathcal{G}})$ be a set that intersects $E(\bar{B}_i)$ for each nonempty \bar{B}_i . Then U is a transversal of $\bar{\mathcal{G}}$ and $|U| \geq \tau(\bar{\mathcal{G}}) = r$.

Proof. Clearly, U is a transversal of $\bar{\mathcal{G}}$; thus $|U| \geq \tau(\bar{\mathcal{G}})$. Proposition 7.10 states that $\tau(\bar{\mathcal{G}}) = \tau(\bar{\mathcal{G}})$. \square

Claim 3. Let U, U' be distinct transversals of $\bar{\mathcal{G}}$. If $\tau(\bar{\mathcal{G}}) = |U| = |U'|$, then $|U - U'| \geq 2$.

Proof. Proposition 7.10 implies that U and U' are minimum transversals of $\bar{\mathcal{G}}$. Hence, $U, U' \in \bar{\mathcal{H}}$. The result now follows from Proposition 7.9. \square

Claim 4. None of the \bar{B}_i is empty.

Proof. Let U be a minimum cardinality set that intersects $E(\bar{B}_i)$ for each nonempty \bar{B}_i . Since $r \geq 3$, it follows from Claim 2 that at most one of the \bar{B}_i can be empty.

Assume for a contradiction that one of the \bar{B}_i , say \bar{B}_4 , is empty. It follows from Claim 2 and the choice of U that each of $E(\bar{B}_1), E(\bar{B}_2), E(\bar{B}_3)$ are pairwise disjoint. (Otherwise U contains an element common to at least 2 of $E(\bar{B}_1), E(\bar{B}_2), E(\bar{B}_3)$ and $|U| \leq 2$.) If $|E(\bar{B}_1)| > 1$, then let t_1, t'_1 be distinct elements of $E(\bar{B}_1)$. Let $t_2 \in E(\bar{B}_2)$ and $t_3 \in E(\bar{B}_3)$. Then $U = \{t_1, t_2, t_3\}$ and $U' = \{t'_1, t_2, t_3\}$ contradict Claim 3. Thus $|E(\bar{B}_1)| = 1$ and, similarly, $|E(\bar{B}_2)| = |E(\bar{B}_3)| = 1$. As $|E_1| > 3$ and $|E_2| > 3$, $\bar{B}_1, \bar{B}_2, \bar{B}_3$ are not all E_1 -blocks and not all E_2 -blocks. Thus without loss of generality we may assume \bar{B}_1, \bar{B}_2 are E_1 -blocks and \bar{B}_3 is an E_2 -block. Let t_1 be any element in $E_1 - E(\bar{B}_1) - E(\bar{B}_2)$ and t_2 be the unique element in $E(\bar{B}_3)$. Then the column of $A(\bar{\mathcal{G}})$ indexed by t_1 is included in the column of $A(\bar{\mathcal{G}})$ indexed by t_2 , a contradiction to Proposition 7.6. \square

First consider the case where every \bar{B}_i is an E_1 -block. Suppose that no two $E(\bar{B}_i)$ intersect. Then $A(\bar{\mathcal{G}})$ has four columns that add up to the vector of all ones. By Theorem 7.4, each of these columns has s ones and therefore $n = 4s$. Furthermore, the four elements that index these columns form a transversal of $\bar{\mathcal{G}}$ and therefore $r \leq 4$ (see Claim 2). This contradicts Theorem 7.4 stating that $rs > n$. Thus two $E(\bar{B}_i)$ intersect, say \bar{B}_1 and \bar{B}_2 . For otherwise $n = 4s$, a contradiction to $rs > n$. Let t be any element of $E(\bar{B}_1) \cap E(\bar{B}_2)$ and let g_3 (resp., g_4) be any element of $E(\bar{B}_3)$ (resp., $E(\bar{B}_4)$). Let $U = \{t, g_3, g_4\}$. It follows from Claim 2 that $r = 3$. It follows from Claim 3 that each of $E(\bar{B}_3)$ and $E(\bar{B}_4)$ have cardinality one, and $E(\bar{B}_1) \cap E(\bar{B}_2)$ contains a unique element e . Since there are no dominated columns in $A(\bar{\mathcal{G}})$ we have that $E(\bar{B}_1) - \{e\} = E(\bar{B}_2) - \{e\} = \emptyset$. Thus $|E_1| \leq 3$, a contradiction to the hypothesis that the 3-separation is strict.

Now consider the case where \bar{B}_1, \bar{B}_2 are E_1 -blocks and \bar{B}_3, \bar{B}_4 are E_2 -blocks. Suppose there exists $e \in E(\mathcal{H})$ that is not in any of $E(\bar{B}_i)$ for $i \in \{1, \dots, 4\}$. Assume without loss of generality that $e \in E_1$. Then column e of $A(\bar{\mathcal{G}})$ is included in the union of any two columns $f_1 \in E(\bar{B}_3)$ and $f_2 \in E(\bar{B}_4)$, a contradiction to Proposition 7.6. Thus every element of $E(\mathcal{H})$ is in $E(\bar{B}_i)$ for some $i \in \{1, \dots, 4\}$. Suppose there is $e \in E(\bar{B}_1) \cap E(\bar{B}_2)$. Let $U = \{e, f_1, f_2\}$. Then Claim 2 implies $r = 3$ and Claim 3 implies that $|E(\bar{B}_3)| = |E(\bar{B}_4)| = 1$. Hence $|E_2| = 2$, a contradiction. Thus E_1 is partitioned into $E(\bar{B}_1), E(\bar{B}_2)$, and E_2 is partitioned into $E(\bar{B}_3), E(\bar{B}_4)$. If $r = 4$, then we can use Claim 3 to show that for each $i \in \{1, \dots, 4\}$, $|E(\bar{B}_i)| = 1$, a contradiction as then $|E_1| = |E_2| = 2$. Thus $r = 3$ and let $T = \{u, v, w\}$ be a minimum transversal of $\bar{\mathcal{G}}$. Suppose both $u, v \in E(\bar{B}_i)$ for some $i \in \{1, \dots, 4\}$, say $i = 3$. If $T \subseteq E_2$, then $w \in E(\bar{B}_4)$, as T is a transversal. It then follows that T intersects all sets of \bar{B}_3 with even parity, a contradiction as \mathcal{H} is binary. Thus we may assume $w \in E(\bar{B}_1)$ and w intersects all sets in \bar{B}_4 . It follows that, for any $x \in E(\bar{B}_2), y \in E(\bar{B}_3)$, the set $\{w, x, y\}$ is a transversal of $\bar{\mathcal{G}}$, a contradiction to Claim 3. Hence for any transversal $T = \{u, v, w\}$ each element of T is in a different $E(\bar{B}_i)$. We may assume $u \in E(\bar{B}_1), v \in E(\bar{B}_3), w \in E(\bar{B}_4)$. It follows that for any $x \in E(\bar{B}_1), \{x, v, w\}$ is a transversal and thus by Claim 3 $E(\bar{B}_1)$ contains a unique element t . Since $|E_1| > 2$, we cannot have a transversal $u \in E(\bar{B}_2), v \in E(\bar{B}_3), w \in E(\bar{B}_4)$, as this would imply $|E(\bar{B}_2)| = 1$. Hence every minimum transversal contains t , a contradiction to Theorem 7.4.

Finally, consider the case where $\bar{B}_1, \bar{B}_2, \bar{B}_3$ are E_1 -blocks and \bar{B}_4 is an E_2 -block. Note that every $t \in E_1$ is in some $E(\bar{B}_i)$ for $i \in \{1, 2, 3\}$. Otherwise the corresponding column t of $A(\bar{\mathcal{G}})$ is dominated by any column $t' \in E(\bar{B}_4)$. Suppose there is $t \in E(\bar{B}_i) - E(\bar{B}_j) - E(\bar{B}_k)$, where i, j, k are distinct elements in $\{1, 2, 3\}$. Proposition 7.7 states that there exist three sets of $\bar{\mathcal{G}}$ that intersect exactly in t . This implies $|E(\bar{B}_i)| =$

1. Now since $E(\bar{B}_j) \neq E(\bar{B}_k)$, there is a column in, say, $E(\bar{B}_j) - E(\bar{B}_i) - E(\bar{B}_k)$. Thus $|E(\bar{B}_j)| = 1$. Similarly, $|E(\bar{B}_k)| = 1$, a contradiction to $|E_1| > 3$. Thus $E(\bar{B}_i) \subseteq E(\bar{B}_j) \cup E(\bar{B}_k)$ for all distinct $i, j, k \in \{1, 2, 3\}$ and therefore either (1) for some distinct $i, j, k \in \{1, 2, 3\}$, $E(\bar{B}_j), E(\bar{B}_k)$ is a partition of $E(\bar{B}_i)$ or (2) $E(\bar{B}_i) \cap E(\bar{B}_j) \neq \emptyset$ for each distinct $i, j \in \{1, 2, 3\}$. By considering sets U containing one element of $E(\bar{B}_4)$ and intersecting each of $E(\bar{B}_1), E(\bar{B}_2)$, and $E(\bar{B}_3)$ we can use Claim 3 to show that $|E_1| \leq 2$ in case (1) and $|E_1| \leq 3$ in case (2), a contradiction. \square

7.2. Parts and minors. In this section, we prove Lemma 7.2. Consider the matroid with exactly three elements 1, 2, 3 which form a circuit C_0 . Let I_0, I_1 be disjoint subsets of C_0 . We say that a signed matroid (N, Γ) is a *fat triangle* (I_0, I_1) if $\Gamma = I_1$ and N is obtained from C_0 by adding a parallel element for every $i \in I_0 \cup I_1$. Let (M, Σ) be a signed binary matroid with a circuit $C_0 = \{1, 2, 3\}$, where $C_0 \cap \Sigma = \emptyset$, and let $i \in C_0$. A circuit C_i of M is a *simple circuit* of type i if $C_i \cap C_0 = \{i\}$ and $|C_i \cap \Sigma| = 1$. We say that a cocircuit D has a *small intersection* with a simple circuit C if either $D \cap C = \emptyset$ or $|D \cap C| = 2$ and the unique element in $C \cap \Sigma$ is in D .

LEMMA 7.13. *Let (M, Σ) be a signed binary matroid with a circuit $C_0 = \{1, 2, 3\}$ such that $C_0 \cap \Sigma = \emptyset$.*

- (1) *Let $I \subseteq C_0$ be such that for all $i \in I$ there is a simple circuit C_i of type i . Suppose for all distinct $i, j \in C_0$ we have a cocircuit D_{ij} , where $D_{ij} \cap C_0 = \{i, j\}$ and D_{ij} has a small intersection with the simple circuits in $\{C_t : t \in I\}$. Then the fat triangle (\emptyset, I) is a minor of (M, Σ) .*
- (2) *Let C_1 be a simple circuit of type 1. Suppose we have a cocircuit D_{12} , where $D_{12} \cap C_0 = \{1, 2\}$ and D_{12} has a small intersection with C_1 . If $C_1 - \{1\} \cup \{2\}$ is dependent, then $C_1 - \{1\} \cup \{2\}$ contains an odd circuit using 2 and the fat triangle $(\{3\}, \{2\})$ is a minor of (M, Σ) .*
- (3) *Suppose for each $i = 1, 2$ we have a simple circuit C_i of type i . Suppose we have a cocircuit D_{12} , where $D_{12} \cap C_0 = \{1, 2\}$ and D_{12} has a small intersection with C_1 and C_2 . If both $C_1 - \{1\} \cup \{2\}$ and $C_2 - \{2\} \cup \{1\}$ are independent, then the fat triangle $(\emptyset, \{1, 2\})$ is a minor of (M, Σ) .*

Proof. Throughout the proof i, j, k will denote distinct elements of C_0 .

Let us prove (1). For each $i \in I$ let f_i be the unique element in $C_i \cap \Sigma$. For each D_{jk} either $D_{jk} \cap C_i = \emptyset$ or $D_{jk} \cap C_i = \{f_i, g_i\}$, where g_i is an element not in Σ . Let E_0 be the set of elements in C_0 or in any of C_i , where $i \in I$. Observe that

- (a) If g_i exists, then f_i is in each of D_{12}, D_{13}, D_{23} and g_i is in D_{jk} but not D_{ij}, D_{ik} ;
- (b) If g_i does not exist but f_i does, then f_i is in D_{ij}, D_{ik} but not in D_{jk} .

Define $\Gamma := (\Sigma \triangle D_{12} \triangle D_{13} \triangle D_{23}) \cap E_0$. Observe that (a) and (b) imply, respectively, (a') and (b').

- (a') If g_i exists, then $f_i \notin \Gamma$ and $g_i \in \Gamma$.
- (b') If g_i does not exist, then $f_i \in \Gamma$.

Let (N, Γ) be the minor of (M, Σ) obtained by deleting the elements not in E_0 and then contracting the elements not in $C_0 \cup \Gamma$. It follows from (a') and (b') that if C_0 is a circuit, then (N, Γ) is the fat triangle (\emptyset, I) . Otherwise some element $i \in C_0$ is a loop of N , say $i = 1$. Then there is a circuit C of M such that $C \subseteq E_0, C \cap C_0 = \{1\}$, and $C \cap \Gamma = \emptyset$. Clearly, $C \cap C_0$ does not intersect D_{12} and D_{23} with the same parity. Consider any $e \in C - C_0$ such that e is in some cocircuit D_{ij} . Since $e \notin \Gamma$, it follows from (a') and (b') that $e = f_i$ for some $i \in I$ and that g_i exists. However, then (a) implies that $e \in D_{12} \cap D_{13} \cap D_{23}$. It follows that C cannot intersect D_{12} and D_{23} with the same parity, a contradiction.

Let us prove (2). Let f be the unique element in $C_1 \cap \Sigma$. By hypothesis there is a circuit C in $C_1 - \{1\} \cup \{2\}$. Since C_1 is a circuit $2 \in C$. Since D_{12} has a small intersection $C_1 \cap D_{12} = \{1, f\}$. It follows that $C \cap D_{12} = \{2, f\}$. Let C' be the circuit using 3 in $C_1 \triangle C \triangle C_0$. Since C_0 is a circuit, 3 is not a loop; hence $C' - \{3\}$ contains at least one element, say g . Let $(N, \Gamma) = (M, \Sigma) \setminus (E(M) - C_0 - C_1) / (C_1 - \{f, g\})$. Observe that $\{2, f\}$ is an odd cycle of (N, Γ) and that $\{3, g\}$ and C_0 are even cycles of (N, Γ) . Hence, if C_0 is a circuit of N , then (N, Γ) is the fat triangle $(\{3\}, \{2\})$. Because D_{12} is a cocircuit of M , $\{1, 2, f\}$ is a cocycle of N ; in particular, 1, 2, f are not loops. If 3 is a loop of N , then there is a circuit $S \subseteq C_1 - \{1, 2, f, g\} \cup \{3\}$ of (M, Σ) . However, $C' \triangle S$ is a cycle and $C' \triangle S \subset C_1$, a contradiction as C_1 is a circuit.

Let us prove (3). Let M' be obtained from M by deleting all elements not in $C_0 \cup C_1 \cup C_2$ and let $\Sigma' := (\Sigma \triangle D_{12}) \cap E(M')$. Since D_{12} has a small intersection with C_1 and C_2 we have $\Sigma' = \{1, 2\}$. Then $(M', \{1, 2\})$ is a signed minor of (M, Σ) . Choose a minor N of M' which is minimal and satisfies the following properties:

- (i) C_0 is a circuit of N .
- (ii) For $i = 1, 2$ there exist circuits C_i of N such that $C_i \cap C_0 = \{i\}$.
- (iii) $C_1 - \{1\} \cup \{2\}$ is independent.
- (iv) $C_2 - \{2\} \cup \{1\}$ is independent.

Note that by hypothesis M satisfies properties (i)–(iv) and thus so does M' . Hence N is well defined. We will show that $|C_1| = |C_2| = 2$ in N . Then $(N, \{1, 2\})$ is a minor of (M, Σ) and after resigning on the cocircuit containing 1, 2 we obtain the fat triangle $(\emptyset, \{1, 2\})$. There is no circuit $S \subseteq C_1 - \{1\} \cup \{3\}$ of N , for otherwise there exists a cycle $C_1 \triangle S \triangle C_0 \subseteq C_1 - \{1\} \cup \{2\}$, a contradiction with (iii). Hence, we have the following claims.

Claim 1. $C_1 - \{1\} \cup \{3\}$ is independent.

Claim 2. $C_1 \cap C_2 = \emptyset$.

Proof. Otherwise define $N' := N / (C_1 \cap C_2)$. Note that N' satisfies (ii)–(iv). Suppose (i) does not hold for N' ; i.e., C_0 is a cycle but not a circuit of N' . Then 3 is a loop of N' . Thus there is $S \subseteq C_1 \cap C_2$ such that $S \cup \{3\}$ is a circuit of N , contradicting Claim 1. \square

Assume for a contradiction $|C_i| > 2$ for some $i \in \{1, 2\}$, say $i = 1$.

Claim 3. There exists a circuit $S \subseteq C_1 \cup C_2 - \{1, 2\}$ of N .

Proof. Let $e \in C_1 - \{1\}$ and consider $(N', \Gamma') := (N, \Gamma) / e$. Suppose C_0 is not a circuit of N' . Then 2 or 3 is a loop of N . However, then either $\{2, e\}$ or $\{3, e\}$ is a circuit of N . In the former case it contradicts (iii); in the latter it contradicts Claim 1. Hence (i) holds for N' . Trivially (iii) holds for N' as well. Suppose (ii) does not hold; then C_2 is not a circuit of N' . It implies there exists a circuit $S \subseteq C_2 \cup \{e\} - \{2\}$ of N . Then S is the required circuit. Suppose (iv) does not hold. Then there is a circuit $S \subseteq C_2 \cup \{e, 1\} - \{2\}$ of N , and $S \triangle C_1$ contains the required circuit. \square

Let S be the circuit in the previous claim. Since C_1, C_2 are circuits, $S \cap C_1, S \cap C_2$ are nonempty. Let C'_2 be the circuit in $C_2 \triangle S$ which uses 2. Note that $N \setminus (E(N) - C_0 - C_1 - C'_2)$ satisfies properties (i)–(iii) using C'_2 instead of C_2 . Thus, by minimality, (iv) is not satisfied for C'_2 ; i.e., $C'_2 - \{2\} \cup \{1\}$ contains a circuit C'_1 . Since C'_2 is a circuit, $1 \in C'_1$. By the same argument as above, (iii) is not satisfied for C'_1 ; i.e., $C'_1 - \{1\} \cup \{2\}$ contains a circuit C''_2 using 2. Since $C''_2 \subseteq C'_2$ it follows that $C''_2 = C'_2$ (since C'_2 is a circuit). Therefore, $C'_1 = C'_2 - \{2\} \cup \{1\}$. However, the cycle $C'_1 \triangle C'_2 = \{1, 2\}$ contradicts the fact that C_0 is a circuit. \square

LEMMA 7.14. *Let (M, Σ) be an ideal signed binary matroid with a circuit $C_0 = \{1, 2, 3\}$, where $C_0 \cap \Sigma = \emptyset$. Suppose we have $p : E(M) \rightarrow Q_+$ such that the cut*

condition is satisfied. Then there exists $p' : E(M) \rightarrow Q_+$ which satisfies the following properties: (i) p' satisfies the cut condition; (ii) $p'(e) = p(e)$ for all $e \notin C_0$; and (iii) $p'(i) + p'(j) \leq p(i) + p(j)$ for all distinct $i, j \in C_0$. Let $I = \{i \in C_0 : p'(i) > 0\}$. There is a Σ -flow $\eta : \Omega_\Sigma \rightarrow Q_+$ with costs p' . Moreover, either

- (1) the fat triangle (\emptyset, I) is a signed minor of (M, Σ) and
- (2) $|C \cap C_0| \leq 1$ for all odd circuits C such that $\eta(C) > 0$,

or after possibly relabeling elements of C_0 we have $p'(3) = 0$ and $p'(2) \leq p(2) + p(3)$. Moreover,

- (3) the fat triangle $(\{3\}, \{2\})$ is a signed minor of (M, Σ) and
- (4) for all odd circuits C with $\eta(C) > 0$ and $C \cap C_0 \neq \emptyset$ either $C \cap C_0 = \{2\}$ or $C \cap C_0 = \{1\}$ and $C - \{1\} \cup \{2\}$ contains an odd circuit using 2.

Proof.

Claim 1. We can assume that there exists $p' : E(M) \rightarrow Q_+$ such that properties (i)–(iii) hold. For distinct $i, j \in C_0$ let α_{ij} be the minimum of $p'(D - \Sigma) - p'(D \cap \Sigma)$, where $D \cap C_0 = \{i, j\}$. We then have (after possibly relabeling the elements of C_0) the following cases: either (a) $\alpha_{12} = \alpha_{13} = \alpha_{23} = 0$ or (b) $p'(3) = 0, p'(2) \leq p(2), p'(1) \leq p(1) + p(3)$, and $\alpha_{12} = 0$.

Proof. Choose $p' : E(M) \rightarrow Q_+$ which minimizes $p'(C_0)$ and which satisfies the following properties: The cut condition holds for p' ; $p'(e) = p(e)$ for all $e \notin C_0$; and $p'(i) \leq p(i)$ for all $i \in C_0$. Clearly, (i)–(iii) holds for p' . Suppose (a) does not hold. Then we may assume (after relabeling) that $\alpha_{23} > 0$ and that $p'(3) \leq p'(2)$.

First consider the case where $\alpha_{12} > 0$. Then 2 is in no tight cocircuit; it follows from the choice of p' that $p'(2) = 0$. Hence $p'(3) = 0$. Suppose $\alpha_{13} > 0$; then $p'(1) = 0$. Therefore for all circuits C such that $\eta(C) > 0$ we have $C \cap C_0 = \emptyset$ and (2) holds. Moreover, (1) is satisfied since $(M, \Sigma) \setminus (E(M) - C_0)$ is the (\emptyset, \emptyset) fat triangle. Thus we may assume $\alpha_{13} = 0$. However, by relabeling 2 and 3 we satisfy (b).

Hence we can assume $\alpha_{12} = 0$. If $\alpha_{13} > 0$, then 3 is in no tight cocircuit; thus $p'(3) = 0$, and (b) holds. Thus we may assume $\alpha_{13} = 0$. Let $\epsilon = \min\{\alpha_{23}/2, p'(3)\}$. Let $\hat{p} : E(M) \rightarrow Q_+$ be defined as follows: $\hat{p}(e) = p'(e)$ if $e \notin C_0$ and $\hat{p}(1) = p'(1) + \epsilon, \hat{p}(2) = p'(2) - \epsilon, \hat{p}(3) = p'(3) - \epsilon$. Note that (i)–(iii) hold for \hat{p} . Suppose $\epsilon = p'(3)$. Then $\hat{p}(3) = 0$, and (b) holds with \hat{p} since $\hat{p}(2) \leq p'(2) \leq p(2)$ and $\hat{p}(1) = p'(1) + \epsilon \leq p(1) + p(3)$. Thus we may assume $\epsilon = \alpha_{32}/2$. Then for each distinct $i, j \in C_0$ there is a cocircuit D , where $D \cap C_0 = \{i, j\}$, which is tight for \hat{p} . It follows that (a) holds. \square

Throughout the proof i, j, k will denote distinct elements of C_0 . Let p' be the costs given in Claim 1. Since (M, Σ) is ideal, Corollary 4.4(i) implies that there is a Σ -flow, $\eta : \Omega_\Sigma \rightarrow Q_+$ for M with costs p' . Let D_{ij} be the cocircuits of M for which $D_{ij} \cap C_0 = \{i, j\}$ and $p(D_{ij} - \Sigma) - p(D_{ij} \cap \Sigma) = \alpha_{ij}$.

First consider case (a) of Claim 1; i.e., D_{ij} is tight for all distinct $i, j \in C_0$. We will show that (1) and (2) hold. Let C be any circuit with $\eta(C) > 0$. Then $|C \cap \Sigma| = 1$. Suppose there is an element i in $C_0 \cap C$. Proposition 4.3 states that $C \cap D_{ij} = \{i, f\}$ and $C \cap D_{ik} = \{i, f\}$, where f is the unique element in $C \cap \Sigma$. Thus $C \cap C_0 = \{i\}$ and (2) holds. Every element $i \in C_0$ is in a tight cocircuit; thus if $p'(i) > 0$, then there is a circuit C_i with $i \in C_i$ and $\eta(C_i) > 0$. Moreover, (2) implies that C_i is a simple circuit of type i . Proposition 4.3 implies that D_{12}, D_{13}, D_{23} all have small intersections with each of the simple circuits. Then (1) follows from Lemma 7.13(1).

Consider case (b) of Claim 1, i.e., $p'(3) = 0, p'(2) \leq p(2), p'(1) \leq p(1) + p(3)$ and $\alpha_{12} = 0$.

Claim 2. Let C be a circuit with $\eta(C) > 0$. If $i \in C \cap \{1, 2\}$, then C_i is a simple

circuit of type i .

This follows from the fact that $3 \notin C$ (as $p'(3) = 0$) and that $|C \cap \{1, 2\}| = 1$ (because of Proposition 4.3 and the fact that D_{12} is tight). The case where $p'(i) = 0$ for all $i \in C_0$ has already been considered (see the proof of Claim 1). Suppose for some $i \in \{1, 2\}$, $p'(3 - i) = 0$. Then $p'(i) > 0$ and let f be the unique element in $C_i \cap \Sigma$. The minor $(M, \Sigma) \setminus (E(M) - C_0 - C_i) / (C_i - \{i, f\})$ is the fat triangle $(\emptyset, \{i\})$ and both (1) and (2) hold. Thus $p'(1) > 0, p'(2) > 0$. Suppose now for all $i \in \{1, 2\}$ there exists a circuit C_i with $\eta(C_i) > 0$ and $i \in C_i$ such that $C_i - \{i\} \cup \{3 - i\}$ is independent. Claim 2 states that these circuits are simple circuits of type i . Then (2) holds and Lemma 7.13(3) implies that (M, Σ) contains the fat triangle $(\emptyset, \{1, 2\})$, i.e., (1) holds. Thus we may assume, for some $i \in \{1, 2\}$, that for all circuits C_i such that $\eta(C_i) > 0$ and $i \in C_i$, $C_i - \{i\} \cup \{3 - i\}$ is dependent. If $i = 2$, interchange the labels 2 and 1. Since we had $p'(1) \leq p(1) + p(3)$ we get in that case $p'(2) \leq p(2) + p(3)$. Otherwise (if $i = 1$) we have $p'(2) \leq p(2) \leq p(2) + p(3)$. Lemma 7.13(2) implies that for all circuits C_1 with $\eta(C_1) > 0$ and $1 \in C_1$, $C_1 - \{1\} \cup \{2\}$ contains an odd circuit using 1 and that (M, Σ) contains the fat triangle $(\{3\}, \{2\})$ as a minor. Together with Claim 2 this implies that (3) and (4) hold. \square

We are now ready for the proof of the main lemma.

Proof of Lemma 7.3. Since M has a strict 3-separation, $M = M_1 \otimes_3 M_2$, where $C_0 = E(M_1) \cap E(M_2)$ is a triangle. Throughout this proof i, j, k will denote distinct elements of C_0 . Recall that $F \subseteq E_1$. Let α_{ij}^1 denote the smallest value of

$$(*) \quad p(D_{ij} - F - C_0) - p(D_{ij} \cap F),$$

where D_{ij} is some cocircuit of M_1 with $D_{ij} \cap C_0 = \{i, j\}$. Expression (*) gives the difference between the sum of the capacity elements and the sum of the demand elements in D_{ij} , excluding the marker C_0 . Denote by D_{ij}^1 the cocircuit for which the minimum is attained in (*). Let α_{ij}^2 denote the smallest value of $p(D_{ij} - C_0)$, where D_{ij} is some cocircuit of M_2 with $D_{ij} \cap C_0 = \{i, j\}$. In what follows, we let D_{ij}^2 denote the cocircuit for which $p(D_{ij}^2 - C_0) = \alpha_{ij}^2$. For each $i \in C_0$ define

$$\beta_i = \frac{1}{2}(\alpha_{ij}^2 + \alpha_{ik}^2 - \alpha_{jk}^2).$$

Claim 1. $\beta_i \geq 0$ for all $i \in C_0$.

Proof. We have $\alpha_{ij}^2 + \alpha_{ik}^2 = p(D_{ij}^2 - C_0) + p(D_{ik}^2 - C_0) \geq p((D_{ij}^2 \Delta D_{ik}^2) - C_0) \geq \alpha_{jk}^2$. Thus $(\alpha_{ij}^2 + \alpha_{ik}^2) - \alpha_{jk}^2 \geq 0$ and $\beta_i \geq 0$. \square

Claim 2. $\alpha_{ij}^1 + \alpha_{ij}^2 \geq 0$.

Proof. $\alpha_{ij}^1 + \alpha_{ij}^2 = p(D_{ij}^1 - F - C_0) - p(D_{ij}^1 \cap F) + p(D_{ij}^2 - C_0) = p((D_{ij}^1 \Delta D_{ij}^2) - F) - p((D_{ij}^1 \Delta D_{ij}^2) \cap F)$. The last expression is nonnegative since the cut condition holds for (M, F, p) . \square

Claim 3. (M_1, F) is a signed minor of (M, F) .

Proof. Theorem 5.2 implies that M_1 is a minor of M obtained by contracting and deleting elements in E_2 only. Since $F \subseteq E_1$ the result follows. \square

Define $p_1 : E(M_1) \rightarrow \mathbb{Q}_+$ as follows: $p_1(e) = p(e)$ for all $e \in E_1$ and $p_1(i) = \beta_i$ for all $i \in C_0$.

Claim 4. The cut condition is satisfied for (M_1, F, p_1) .

Proof. Since the cut condition holds for (M, F, p) the cut condition is satisfied for all cocircuits of M_1 disjoint from C_0 . Let D be a cocircuit of M_1 such that $D \cap C_0 = \{i, j\}$. Then $p_1(D - F) - p_1(D \cap F) = p(D - F - C_0) - p(D \cap F) + p_1(i) + p_1(j) \geq$

$\alpha_{ij}^1 + p_1(i) + p_1(j) = \alpha_{ij}^1 + \beta_i + \beta_j = \alpha_{ij}^1 + \alpha_{ij}^2$. It follows from Claim 2 that the previous expression is nonnegative. \square

Claim 3 implies that (M_1, F) is a part of (M, F) , and hence its clutter of odd circuits is ideal. Together with Claim 4 it implies that (M_1, F) and p_1 satisfy the hypothesis of Lemma 7.14. It follows that M_1 is F -flowing with costs p'_1 (where p'_1 is as described in the lemma) and either Case 1 or Case 2 occurs.

Case 1. Statements (1) and (2) hold.

We define $I := \{i \in C_0 : p'_1(i) > 0\}$ and let M'_2 denote $M_2 \setminus (C_0 - I)$.

Claim 5. (M'_2, I) is a signed minor of (M, F) .

Proof. Statement (1) says that the fat triangle (\emptyset, I) is a signed minor of (M_1, F) ; i.e., it is equal to $(M_1, F) \setminus J_d/J_c$ for some $J_d, J_c \subseteq E_1$. Seymour [24] showed that $(M_1 \otimes_3 M_2) \setminus J_d/J_c = (M_1 \setminus J_d/J_c) \otimes_3 M_2$. Thus $(M'_2, I) = (M, F) \setminus J_d/J_c$. \square

Define $p_2 : E(M'_2) \rightarrow \mathbb{Q}_+$ as follows: $p_2(e) = p(e)$ for all $e \in E_2$ and $p_2(i) = p'_1(i)$ for all $i \in I$.

Claim 6. The cut condition is satisfied for (M'_2, I, p_2) .

Proof. It suffices to show the cut condition holds for cocircuits D that intersect C_0 . Suppose $D \cap C_0 = \{i, j\}$. Lemma 7.14 states that $p'_1(i) + p'_1(j) \leq p_1(i) + p_1(j)$. Moreover, $p_1(i) + p_1(j) = \beta_i + \beta_j = \alpha_{ij}^2$. Thus $p_2(D - I) - p_2(D \cap I) = p(D - C_0) - (p'_1(i) + p'_1(j)) \geq \alpha_{ij}^2 - \alpha_{ij}^2 = 0$. \square

Claim 5 implies that (M'_2, I) is a part of (M, F) , and hence its clutter of odd circuits is ideal. It follows from Claim 6 and Corollary 4.4(i) that M'_2 is I -flowing with costs p_2 . Since we can scale p (and hence p'_1 and p_2) we may assume that the F -flow of M_1 satisfying costs p'_1 is a multiset \mathcal{L}^1 of circuits and that the I -flow of M'_2 satisfying costs p_2 is a multiset \mathcal{L}^2 of circuits. Because of statement (2), \mathcal{L}^1 can be partitioned into \mathcal{L}_0^1 and \mathcal{L}_i^1 for all $i \in I$, where $\mathcal{L}_0^1 = \{C \in \mathcal{L}^1 : C \cap C_0 = \emptyset\}$ and $\mathcal{L}_i^1 = \{C \in \mathcal{L}^1 : C \cap C_0 = \{i\}\}$. Because $C \in \mathcal{L}^2$ implies $C \in \Omega_I$, \mathcal{L}^2 can be partitioned into \mathcal{L}_i^2 for all $i \in I$, where $\mathcal{L}_i^2 = \{C \in \mathcal{L}^2 : C \cap C_0 = \{i\}\}$. Since $p_2(i) = p'_1(i)$ for each $i \in I$, $|\mathcal{L}_i^1| \leq |\mathcal{L}_i^2|$ for each $i \in I$. Let us define a collection of circuits of M as follows: Include all circuits of \mathcal{L}_0^1 , and for every $i \in I$ pair each circuit $C_1 \in \mathcal{L}_i^1$ with a different circuit $C_2 \in \mathcal{L}_i^2$ and add to the collection the circuit included in $C_1 \triangle C_2$ that contains the element of F . The resulting collection corresponds to a F -flow of M satisfying costs p .

Case 2. $p'_1(3) = 0, p'_1(2) \leq p(2) + p(3)$ and statements (3) and (4) hold (after possibly relabeling C_0).

Let M'_2 denote $M_2 \setminus 1$. Statement (3) says that the fat triangle $(\{3\}, \{2\})$ is a signed minor of (M_1, F) . Proceeding as in the proof of Claim 5 we obtain the following.

Claim 7. $(M'_2, \{2\})$ is a signed minor of (M, F) .

Define $p_2 : E(M'_2) \rightarrow \mathbb{Q}_+$ as follows: $p_2(e) = p(e)$ for all $e \in E(M_2)$; $p_2(2) = p'_1(2) + p'_1(1)$ and $p_2(3) = p'_1(1)$.

Claim 8. The cut condition is satisfied for $(M'_2, \{2\}, p_2)$.

Proof. Consider first a cocircuit D of M'_2 such that $D \cap C_0 = \{2, 3\}$. Let us check that D does not violate the cut condition. The following expression should be nonnegative: $p_2(D - \{2\}) - p_2(D \cap \{2\}) = p(D - C_0) + p_2(3) - p_2(2) = p(D - C_0) + p'_1(1) - p'_1(1) - p'_1(2) = p(D - C_0) - p'_1(2)$. Lemma 7.14 states that $p'_1(2) \leq p_1(2) + p_1(3) = \beta_2 + \beta_3$. Since $p(D - C_0) \geq \alpha_{23}^2 = \beta_2 + \beta_3$ it follows that $p(D - C_0) - p'_1(2) \geq 0$. Consider a cocircuit D of M'_2 such that $2 \in D$ but $3 \notin D$. Let us check that D does not violate the cut condition. The following expression should be nonnegative: $p_2(D - \{2\}) - p_2(D \cap \{2\}) = p(D - C_0) - p_2(2) = p(D - C_0) - (p'_1(1) + p'_1(2))$. Lemma 7.14

states that $p'_1(1) + p'_1(2) \leq p_1(1) + p_1(2) = \beta_1 + \beta_2$. Since $D \cup \{1\}$ is a cocircuit of M_2 , $p_2(D - C_0) \geq \alpha_{12}^2 = \beta_1 + \beta_2$. It follows that $p(D - C_0) - (p'_1(1) + p'_1(2)) \geq 0$. \square

Claim 7, Claim 8, and Corollary 4.4(i) imply that M'_2 is $\{2\}$ -flowing with costs p_2 . We may assume that the F -flow of M_1 satisfying costs p'_1 is a multiset \mathcal{L}^1 of circuits. Because of statement (4), \mathcal{L}^1 can be partitioned into $\mathcal{L}_0^1, \mathcal{L}_1^1, \mathcal{L}_2^1$, where $\mathcal{L}_0^1 = \{C \in \mathcal{L}^1 : C \cap C_0 = \emptyset\}$, $\mathcal{L}_1^1 = \{C \in \mathcal{L}^1 : C \cap C_0 = \{1\}\}$, $\mathcal{L}_2^1 = \{C \in \mathcal{L}^1 : C \cap C_0 = \{2\}\}$. We may assume that the $\{2\}$ -flow of M'_2 satisfying costs p_2 is a multiset \mathcal{L}^2 of circuits. Since $C \in \mathcal{L}^2$ implies $C \in \Omega_{\{2\}}$ and since $1 \notin E(M'_2)$, \mathcal{L}^2 can be partitioned into $\mathcal{L}_1^2, \mathcal{L}_2^2$, where $\mathcal{L}_1^2 = \{C \in \mathcal{L}^2 : C \cap C_0 = \{2, 3\}\}$ and $\mathcal{L}_2^2 = \{C \in \mathcal{L}^2 : C \cap C_0 = \{2\}\}$.

Claim 9. (i) $|\mathcal{L}_2^2| \leq |\mathcal{L}_2^1|$ and (ii) $|\mathcal{L}_1^2| + |\mathcal{L}_1^1| \leq |\mathcal{L}_1^2| + |\mathcal{L}_2^1|$.

Proof. Let us prove (i). 2 is a demand element for the flow \mathcal{L}^2 ; thus $|\mathcal{L}_2^2| + |\mathcal{L}_1^2| = p_2(2) = p'_1(1) + p'_1(2)$. 3 is a capacity element for the flow \mathcal{L}^2 ; thus $|\mathcal{L}_1^2| \leq p_2(3) = p'_1(1)$. Hence, $|\mathcal{L}_2^2| \geq p'_1(2) \geq |\mathcal{L}_2^1|$ where the last inequality follows from the fact that 2 is a capacity element for the flow \mathcal{L}^1 . Let us prove (ii). $|\mathcal{L}_1^2| + |\mathcal{L}_1^1| \leq p'_1(2) + p'_1(1) = p_2(2) = |\mathcal{L}_2^2| + |\mathcal{L}_2^1|$. \square

Let us define a collection of circuits of M as follows: (a) Include all circuits of \mathcal{L}_0^1 ; (b) pair every circuit $C_1 \in \mathcal{L}_2^1$ with a different circuit $C_2 \in \mathcal{L}_2^2$ —such a pairing exists because of Claim 9(i)—and add to the collection $C_1 \triangle C_2$; (c) pair as many circuits C_1 of \mathcal{L}_1^1 to as many different circuits C_2 of \mathcal{L}_1^2 as possible, and add to the collection $C_1 \triangle C_2$; (d) pair all remaining circuits C_1 of \mathcal{L}_1^1 to circuits of \mathcal{L}_2^2 not already used in (b). Such a pairing exists because of Claim 9(ii). Statement (4) says that $C_1 - \{1\} \cup \{2\}$ contains an odd circuit C'_1 . For every pair C_1, C_2 add to the collection the cycle $C'_1 \triangle C_2$; (e) for each cycle C in the collection keep only the circuit included in C that contains the element of F . The resulting collection corresponds to an F -flow of M satisfying costs p . \square

8. Sufficient conditions for idealness. We will prove Theorem 1.1 in this section, i.e., that a binary clutter is ideal if it has none of the following minors: \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , $b(\mathcal{O}_{K_5})^+$, Q_6^+ , and Q_7^+ . The next result is fairly straightforward.

PROPOSITION 8.1 (Novick and Sebö [20]).

- \mathcal{H} is a clutter of odd circuits of a graph if and only if $u(\mathcal{H})$ is graphic.
- \mathcal{H} is a clutter of T -cuts if and only if $u(\mathcal{H})$ is cographic.

Remark 8.2. The class of clutters of odd circuits and the class of clutters of T -cuts is closed under taking minors.

This follows from the previous proposition, Remark 2.9, and the fact that the classes of graphic and cographic matroid are closed under taking (matroid) minors. We know from Remark 3.4 that $b(Q_6)^+$ (a minor of Q_7^+) is a source of F_7 , and Q_6^+ is a source of F_7^* . Thus Proposition 8.1 implies the following remark.

Remark 8.3. Q_7^+ and Q_6^+ are not clutters of odd circuits or clutters of T -cuts.

We use the following two decomposition theorems.

THEOREM 8.4 (Seymour [24]). *Let M be a 3-connected and internally 4-connected regular matroid. Then $M = R_{10}$ or M is graphic or M is cographic.*

THEOREM 8.5 (Seymour [24, 26]). *Let M be a 3-connected binary matroid with no F_7^* (resp., F_7) minor. Then M is regular or $M = F_7$ (resp., F_7^*).*

COROLLARY 8.6. *Let \mathcal{H} be a binary clutter such that $u(\mathcal{H})$ has no F_7^* minor. If \mathcal{H} is 3-connected and internally 4-connected, then \mathcal{H} is one of $b(Q_7), \mathcal{L}_{F_7}, b(Q_6)^+,$ or one of the 6 lifts of R_{10} , or a clutter of odd circuits or a clutter of T -cuts.*

Proof. Since \mathcal{H} is 3-connected, $u(\mathcal{H})$ is 3-connected. Therefore, by Theorem 8.5, $u(\mathcal{H})$ is regular or $u(\mathcal{H}) = F_7$. In the latter case, Remark 3.4 implies that \mathcal{H} is one of $b(Q_7), \mathcal{L}_{F_7}, b(Q_6)^+$. Thus we can assume that $u(\mathcal{H})$ is regular. By hypothesis, $u(\mathcal{H})$ is

internally 4-connected and therefore, by Theorem 8.4, $u(\mathcal{H}) = R_{10}$ or $u(\mathcal{H})$ is graphic or $u(\mathcal{H})$ is cographic. Now the corollary follows from Proposition 8.1 and Remark 3.4. \square

We are now ready for the proof of the main result of this paper.

Proof of Theorem 1.1. We need to prove that if \mathcal{H} is nonideal, then it contains \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , $b(\mathcal{O}_{K_5})$, Q_6^+ , or Q_7^+ as a minor. Without loss of generality we may assume that \mathcal{H} is mni. It follows from Remark 5.3 and Propositions 6.1 and 7.1 that \mathcal{H} is 3-connected and internally 4-connected. First consider the case where $u(\mathcal{H})$ has no F_7^* minor. Then, by Corollary 8.6 either (i) \mathcal{H} is one of $b(Q_7)$, \mathcal{L}_{F_7} , $b(Q_6)^+$, or (ii) \mathcal{H} is one of the 6 lifts of R_{10} , or (iii) \mathcal{H} is a clutter of odd circuits, or (iv) \mathcal{H} is a clutter of T-cuts. Since \mathcal{H} is mni, it follows from Proposition 3.5 that if (i) occurs then $\mathcal{H} = \mathcal{L}_{F_7}$ and if (ii) occurs, then $\mathcal{H} = b(\mathcal{O}_{K_5})$. If (iii) occurs, then, by Theorem 1.3, $\mathcal{H} = \mathcal{O}_{K_5}$; (iv) cannot occur because of Theorem 1.4.

Now consider the case where $u(\mathcal{H})$ has an F_7^* minor. It follows by Theorem 3.2 that \mathcal{H} has a minor \mathcal{H}_1 or \mathcal{H}_2^+ , where \mathcal{H}_1 is a source of F_7^* and \mathcal{H}_2 is a lift of F_7^* . Proposition 3.1 states that the lifts of F_7^* are the blockers of the sources of F_7 . Remark 3.4 states that the sources of F_7 are $b(Q_7)$, \mathcal{L}_{F_7} , or $b(Q_6)^+$ and that F_7^* has only one source, namely Q_6^+ . This implies that $\mathcal{H}_1 = Q_6^+$ and $\mathcal{H}_2^+ = Q_7^+$ or $b(\mathcal{L}_{F_7})^+$ or $b(b(Q_6^+))^+$. Since $b(\mathcal{L}_{F_7})^+$ has an \mathcal{L}_{F_7} minor and $b(b(Q_6^+))^+$ has a Q_6^+ minor, the proof of the theorem is complete. \square

One can obtain a variation of Theorem 1.1 by modifying Corollary 8.6 as follows: Let \mathcal{H} be a binary clutter such that $u(\mathcal{H})$ has no F_7 minor. If \mathcal{H} is 3-connected and internally 4-connected, then \mathcal{H} is $b(Q_7)$, \mathcal{L}_{F_7} , $b(Q_6^+)$, or one of the 6 lifts of R_{10} or a clutter of odd circuits or a clutter of T-cuts. Following the proof of Theorem 1.1, this yields the following: A binary clutter is ideal if it does not have an \mathcal{L}_{F_7} , \mathcal{O}_{K_5} , $b(\mathcal{O}_{K_5})$, $b(Q_7)$, or $b(Q_6^+)$ minor. However, this result is weaker than Corollary 1.2. Other variations of Theorem 1.1 can be obtained by using Seymour’s splitter theorem [24] which implies, since $u(\mathcal{H})$ is 3-connected and $u(\mathcal{H}) \neq F_7^*$, that $u(\mathcal{H})$ has either S_8 or $AG(3, 2)$ as a minor. Again, by using Theorem 3.2, we can obtain a list of excluded minors that are sufficient to guarantee that \mathcal{H} is ideal.

9. Some additional comments. Corollary 8.6 implies the following result, using the argument used in the proof of Theorem 1.1.

THEOREM 9.1. *Let \mathcal{H} be an ideal binary clutter such that $u(\mathcal{H})$ has no F_7^* minor. If \mathcal{H} is 3-connected and internally 4-connected, then \mathcal{H} is one of $b(Q_7)$, $b(Q_6)^+$, or one of the 5 ideal lifts of R_{10} , or a clutter of odd circuits of a weakly bipartite graph, or a clutter of T-cuts.*

A possible strategy for resolving Seymour’s conjecture would be to generalize this theorem by removing the assumption that $u(\mathcal{H})$ has no F_7^* minor, while allowing in the conclusion the possibility for \mathcal{H} to also be a clutter of T-joins or the blocker of a clutter of odd circuits in a weakly bipartite graph. However, this is not possible as illustrated by the following example.

Let T_{12} be the binary matroid with the following partial representation:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matroid first appeared in [11]. It is self-dual and satisfies the following properties:

- (i) For every element t of T_{12} , T_{12}/t is 3-connected and internally 4-connected.
- (ii) For every element t of T_{12} , T_{12}/t is not regular.

We are indebted to James Oxley (personal communication) for bringing to our attention the existence of the matroid T_{12} and pointing out that it satisfies properties (i) and (ii). Let t be any element of T_{12} and let $\mathcal{H} = \text{Port}(T_{12}, t)$. Because of (i), $T_{12}/t = u(\mathcal{H})$ is 3-connected and internally 4-connected and thus so is \mathcal{H} . Because of (ii), $T_{12}/t = u(\mathcal{H})$ is not graphic or cographic; thus Proposition 8.1 implies that \mathcal{H} is not a clutter of T -cuts and not a clutter of odd circuits. We know from Proposition 2.12 that $b(\mathcal{H}) = \text{Port}(T_{12}^*, t) = \text{Port}(T_{12}, t)$. Thus, $b(\mathcal{H})$ is also 3-connected, internally 4-connected, and \mathcal{H} is not the clutter of T -joins or the blocker of the clutter of odd circuits. However, it follows from the results of Luetolf and Margot [16] that the clutter \mathcal{H} is ideal.

REFERENCES

- [1] R. E. BIXBY, *On the length-width inequality for compound clutters*, J. Combin. Theory Ser. B, 11 (1971) pp. 246–248.
- [2] W. G. BRIDGES AND H. J. RYSER, *Combinatorial designs and related systems*, J. Algebra, 13 (1969), pp. 432–446.
- [3] T. H. BRYLAWSKI, *A decomposition for combinatorial geometries*, Trans. Amer. Math. Soc., 171 (1972), pp. 235–282.
- [4] S. CHOPRA, *Composition for matroids with the Fulkerson property*, Discrete Appl. Math., 62 (1995), pp. 87–101.
- [5] J. EDMONDS AND R. GILES, *A min-max relation for submodular functions on graphs*, Ann. Discrete Math., 1 (1977), pp. 185–204.
- [6] J. EDMONDS AND E. L. JOHNSON, *Matching, Euler tours and the Chinese postman*, Math. Programming, 5 (1973), pp. 88–124.
- [7] D. R. FULKERSON, *Blocking and anti-blocking pairs of polyhedra*, Math. Programming, 1 (1971), pp. 168–194.
- [8] A. M. H. GERARDS, *Graphs and Polyhedra, Binary Spaces and Cutting Planes*, CWI Tract 73, Mathematische Centrum, Amsterdam, 1990.
- [9] A. M. H. GERARDS, *Multi-commodity flows and polyhedra*, CWI Quarterly, 6 (1993), pp. 281–296.
- [10] B. GUENIN, *A characterization of weakly bipartite graphs*, J. Combin. Theory Ser. B, 83 (2001), pp. 112–168.
- [11] S. R. KINGAN, *A generalization of a graph result of D.W. Hall*, Discrete Math., 173 (1997), pp. 129–135.
- [12] M.-K. KWAN, *Graphic programming using odd or even points*, Acta Math. Sinica, 10 (1960), pp. 263–266 (in Chinese).
- [13] A. LEHMAN, *A solution of the Shannon switching game*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 687–725.
- [14] A. LEHMAN, *On the width-length inequality*, Math. Programming, 17 (1979), pp. 403–417.
- [15] A. LEHMAN, *The width-length inequality and degenerate projective planes*, in Polyhedral Combinatorics, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 1, W. Cook and P.D. Seymour, eds., AMS, Providence, RI, 1990, pp. 101–105.
- [16] C. LUETOLF AND F. MARGOT, *A catalog of minimally nonideal matrices*, Math. Methods Oper. Res., 47 (1998), pp. 221–241.
- [17] P. NOBILI AND A. SASSANO, *The anti-join composition and polyhedra*, Discrete Math., 119 (1993), pp. 141–166.
- [18] P. NOBILI AND A. SASSANO, *Polyhedral properties of clutter amalgam*, SIAM J. Discrete Math., 6 (1993), pp. 139–151.
- [19] P. NOBILI AND A. SASSANO, *Composition operations for clutters and related polyhedra*, Methods Oper. Res., 62 (1990), pp. 235–247.
- [20] B. NOVICK AND A. SEBÖ, *On combinatorial properties of binary spaces*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 920, Springer-Verlag, Berlin, 1995, pp. 212–227.
- [21] J. G. OXLEY, *Matroid Theory*, Oxford University Press, New York, 1992.

- [22] M. W. PADBERG, *Lehman's forbidden minor characterization of ideal 0-1 matrices*, Discrete Math., 111 (1993), pp. 409–420.
- [23] P. D. SEYMOUR, *The matroids with the max-flow min-cut property*, J. Combin. Theory Ser. B, 23 (1977), pp. 189–222.
- [24] P. D. SEYMOUR, *Decomposition of regular matroids*, J. Combin. Theory Ser. B, 28 (1980), pp. 305–359.
- [25] P. D. SEYMOUR, *A note on the production of matroid minors*, J. Combin. Theory Ser. B, 22 (1977), pp. 289–295.
- [26] P. D. SEYMOUR, *Matroids and multicommodity flows*, European J. Combin., 2 (1981), pp. 257–290.
- [27] P. D. SEYMOUR, *On Lehman's width-length characterization*, in Polyhedral Combinatorics, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 1, W. Cook and P.D. Seymour, eds., AMS, Providence, RI, 1990, pp. 107–117.
- [28] T. ZASLAVSKY, *Signed graphs*, Discrete Appl. Math., 4 (1982), pp. 47–74.
- [29] T. ZASLAVSKY, *Biased graphs. II. The three matroids*, J. Combin. Theory Ser. B, 51 (1991), pp. 46–72.

DOMINATING PAIR GRAPHS*

JITENDER S. DEOGUN[†] AND DIETER KRATSCH[‡]

Abstract. A pair of vertices of a graph is called a dominating pair if the vertex set of every path between these two vertices is a dominating set of the graph. A graph is a weak dominating pair graph if it has a dominating pair. Further, a graph is called a dominating pair graph if each of its connected induced subgraphs is a weak dominating pair graph. Dominating pair graphs form a class of graphs containing interval, permutation, cocomparability, and asteroidal triple-free graphs.

Our purpose is to study the structural properties of dominating pair graphs. Our main results are a polar theorem for the dominating pairs in weak dominating pair graphs and an existence theorem for minimum cardinality connected dominating sets that induce a simple path in connected dominating pair graphs of diameter not equal to three. Furthermore, we present a forbidden induced subgraph characterization of chordal dominating pair graphs.

Key words. algorithms, graph classes, asteroidal triple-free graphs, dominating pair graphs

AMS subject classification. 05C75

PII. S0895480100367111

1. Introduction. A graph is called a dominating pair graph if each of its connected induced subgraphs has a dominating pair. Similarly, a graph is called a diametral path graph if every connected induced subgraph has a dominating diametral path, where a diametral path is a shortest path of a graph whose length is equal to the diameter of the graph. Diametral path graphs and dominating pair graphs were introduced as models for the design and analysis of networks in an adversarial environment [6, 7]. Both graph classes are natural extensions of the class of asteroidal triple-free graphs for which a long list of structural and algorithmic properties has been developed, in particular by Corneil, Olariu, and Stewart (see, e.g., [3, 4]). Furthermore, well-known graph classes such as interval, permutation, trapezoid, and cocomparability graphs are subclasses of AT-free graphs and thus also of diametral path graphs and dominating pair graphs.

Naturally diametral path graphs as well as dominating pair graphs are good candidates for investigating structural properties similar to those established for asteroidal triple-free graphs. Some structural properties of diametral path graphs are shown in [7].

In this paper, our objective is to investigate structural properties of dominating pair graphs. We are interested in the natural question, Which of the structural properties of asteroidal triple-free graphs can be extended to dominating pair graphs (or even larger classes of graphs)? To anticipate, our main results are the following:

- An interesting polar theorem for dominating pairs in asteroidal triple-free graphs is given in [3]. We shall show that it can be extended to all graphs $G = (V, E)$ having a dominating pair, called weak dominating pair graphs,

*Received by the editors February 2, 2000; accepted for publication (in revised form) March 18, 2002; published electronically May 23, 2002.

<http://www.siam.org/journals/sidma/15-3/36711.html>

[†]Department of Computer Science and Engineering, University of Nebraska at Lincoln, Lincoln, NE 68588-0115 (deogun@cse.unl.edu). This author's research was supported by the Office of Naval Research grant N0014-91-J-1693.

[‡]Université de Metz, Laboratoire d'Informatique Théorique et Appliquée, 57045 Metz Cedex 01, France (kratsch@lita.univ-metz.fr). This research was done while the author was with the Friedrich-Schiller-Universität Jena (Germany) and while he was visiting the University of Nebraska at Lincoln.

if their diameter is sufficiently large: “There are disjoint sets $X \subseteq V$ and $Y \subseteq V$ such that, for all $x, y \in V$, (x, y) is a dominating pair of G if and only if $x \in X$ and $y \in Y$.” (See Theorem 4.1.)

- We present a characterization by forbidden induced subgraphs of those dominating pair graphs that are chordal. This theorem is related to a characterization by forbidden induced subgraphs of interval graphs (i.e. those asteroidal triple-free graphs that are chordal) by Lekkerkerker and Boland [12]. (See Theorem 5.3.)
- Every connected dominating pair graph of diameter not equal to 3 has a PATH-MCDS (i.e., a minimum connected dominating set that induces a path), which almost extends the theorem of [3] stating that every connected asteroidal triple-free graph has a PATH-MCDS. (See Theorem 6.6.)

2. Preliminaries. We present some definitions and properties related to dominating pair graphs. For standard graph theory notations not given here we refer to [1]. For information on special graph classes and their structural properties we refer to [2, 9].

By $G = (V, E)$ we denote a finite, undirected, and simple graph, n denotes the number of vertices, and m denotes the number of edges. $G[W]$ denotes the subgraph of the graph $G = (V, E)$ induced by the vertex set $W \subseteq V$. For any $x \in V$ let $N(x) = \{y : \{x, y\} \in E\}$ be the *open neighborhood* of x and $N[x] = N(x) \cup \{x\}$ the *(closed) neighborhood* of x . Furthermore, $N[A] = \bigcup_{a \in A} N[a]$ for any $A \subseteq V$.

Let $G = (V, E)$ be a graph. A set of vertices $D \subseteq V$ is a *dominating set* of G if every vertex in $V \setminus D$ has a neighbor in D . Furthermore, we say that $X \subseteq V$ *dominates* the set $Y \subseteq V$ if $Y \subseteq N[X]$. Let $W \subseteq V$ be any vertex set. A vertex $y \in V$ is a *private neighbor* of a vertex $x \in W$ with respect to W if x is the only neighbor of y in W , i.e., $N[y] \cap W = \{x\}$. Accordingly, we define $Pr(x, W) := N(x) \setminus N[W \setminus \{x\}]$. Private neighbors with respect to a dominating set D are important for our investigations in this paper.

A sequence of vertices $P = (u = x_0, x_1, \dots, x_k = v)$ in $G = (V, E)$ is called a *path* in G if $\{x_i, x_{i+1}\} \in E$ for all $i \in \{1, 2, \dots, k-1\}$. To identify starting and ending vertices in P , P is also called a *u, v -path*. The path P is *simple* if the vertices x_0, x_1, \dots, x_k are distinct. A simple path $P = (u = x_0, x_1, \dots, x_k = v)$ of G is *chordless* if $G[V(P)] \cong P_{k+1}$. Furthermore, the path $P = (u = x_0, x_1, \dots, x_k = v)$ is said to be *dominating* if its vertex set $V(P) = \{x_0, x_1, \dots, x_k\}$ is a dominating set of G . For $A \subseteq V$ and $B \subseteq V$, an *A, B -path* is a u, v -path with $u \in A$ and $v \in B$. The *distance* between two vertices u and v in G , denoted by $d_G(u, v)$, is the length of a shortest path between u and v . The *diameter* of a graph G is defined as $diam(G) = \max\{d_G(u, v) : u, v \in V\}$. A pair of vertices $u, v \in V$ with $d_G(u, v) = diam(G)$ is called a *diametral pair*, and a shortest path between the vertices of a diametral pair is called a *diametral path*.

DEFINITION 2.1. *A set of three independent vertices x, y, z of a graph G is called an asteroidal triple (AT for short) if for any two of these vertices there exists a path joining them that avoids the (closed) neighborhood of the third. A graph G is asteroidal triple-free (AT-free for short) if G does not contain an AT.*

AT-free graphs form a large class of graphs containing such well-known graph classes as the classes of cocomparability, trapezoid, permutation, and interval graphs. Corneil, Olariu, and Stewart have developed a number of structural and algorithmic properties of AT-free graphs [3, 4]. One of the main theorems of [3] considers dominating pairs.

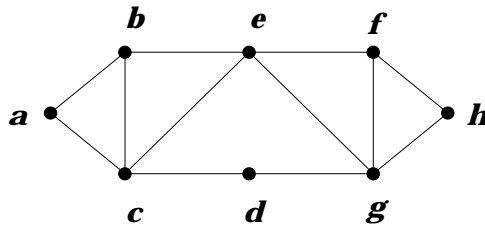


FIG. 2.1. A dominating pair graph of diameter 4 without a diametral DP.

DEFINITION 2.2. A pair (x, y) of vertices of a graph G is a dominating pair (DP for short) if, for every path P between x and y , the vertex set $V(P)$ is a dominating set of G . (It is worth mentioning that $x = y$ is allowed.) (x, y) is a diametral dominating pair (diametral DP for short) of G if (x, y) is a diametral pair and a DP.

THEOREM 2.3 (see [3]). Any connected AT-free graph has a DP. Moreover, any connected AT-free graph has a diametral DP.

Diametral path graphs and dominating pair graphs were introduced in [7].

DEFINITION 2.4. A graph $G = (V, E)$ is a diametral path graph if every connected induced subgraph H of G has a dominating diametral path, i.e., a diametral path P such that the vertex set $V(P)$ is a dominating set of G . A graph $G = (V, E)$ is a dominating pair graph if every connected induced subgraph H of G has a DP.

Every AT-free graph is a dominating pair graph by Theorem 2.3. On the other hand, it is easy to see that not every dominating pair graph is AT-free. (Consider, e.g., C_6 , the chordless cycle on six vertices.) Hence AT-free graphs form a proper subclass of the class of dominating pair graphs. Notice that not every property of AT-free graphs extends to dominating pair graphs: although every connected AT-free graph has a diametral DP by Theorem 2.3, there are connected dominating pair graphs without a diametral DP (see Figure 2.1).

3. Weak dominating pair graphs. It shall turn out that the following graph class is interesting on its own.

DEFINITION 3.1. A graph $G = (V, E)$ is a weak dominating pair graph if G has a DP.

In this section we investigate some structural properties of weak dominating pair graphs. In the next section we establish a polar theorem for weak dominating pair graphs.

Minimal separators are helpful for understanding the structure of weak dominating pair graphs.

DEFINITION 3.2. Let $G = (V, E)$ be a graph. $S \subseteq V$ is a separator of G if $G[V \setminus S]$ is disconnected. A subset $S \subseteq V$ is an x, y -separator for nonadjacent vertices x and y if the removal of S separates x and y into distinct connected components. If no proper subset of the x, y -separator S is itself an x, y -separator, then S is a minimal x, y -separator.

Throughout the paper we make use of the following well-known property (see [9]).

LEMMA 3.3. Let S be a minimal x, y -separator of $G = (V, E)$, and let C_x and C_y be the connected components of $G[V \setminus S]$ containing x and y , respectively. Then every vertex of S has a neighbor in C_x and a neighbor in C_y .

DEFINITION 3.4. Let $G = (V, E)$ be a graph and S a separator of G . Then a component C of $G[V \setminus S]$ is a far component if there is a vertex of C that is not adjacent to all vertices of S .

Lemma 3.5 provides an important tool for the proof of the polar theorem in the next section.

LEMMA 3.5. Let $G = (V, E)$ be a graph with DP (x, y) such that x and y are nonadjacent. Let S be a minimal x, y -separator of G . Then the only possible far components of $G[V \setminus S]$ are C_x and C_y , i.e., the components of $G[V \setminus S]$ containing x and y , respectively.

Proof. Suppose C is a far component of $G[V \setminus S]$ and C is different from C_x and C_y . Then there is a vertex z of C and a vertex $s \in S$ that are nonadjacent. Hence there is an x, y -path avoiding $N[z]$ consisting of an x, s -path with all internal vertices in C_x followed by an s, y -path with all internal vertices in C_y . Hence (x, y) is not a DP, a contradiction. \square

DEFINITION 3.6. Let k be a positive integer. Then an AT $\{a, b, c\}$ of a graph G is said to be k -distant if $d_G(a, b) \geq k$, $d_G(a, c) \geq k$, and $d_G(b, c) \geq k$.

The following lemma provides a useful necessary condition for ATs in weak dominating pair graphs.

LEMMA 3.7. Let $G = (V, E)$ be a weak dominating pair graph. Then G cannot have a 3-distant AT.

Proof. Let (x, y) be a DP and let $\{a, b, c\}$ be a 3-distant AT of G . Consider a shortest x, y -path $P = (x = x_0, x_1, \dots, x_t = y)$ in G . $V(P) = \{x_0, x_1, \dots, x_t\}$ is a dominating set of G , since (x, y) is a DP. Thus every vertex of G has a neighbor in the path P . Let $w = x_i$ be the leftmost vertex of P which is adjacent to a vertex of $\{a, b, c\}$, and let $z = x_j$ be the rightmost vertex of P adjacent to a vertex of $\{a, b, c\}$. Since $\{a, b, c\}$ is 3-distant no two of the three vertices of $\{a, b, c\}$ have a common neighbor. Therefore $j > i + 1$.

If there is a vertex in $\{a, b, c\}$, say vertex a , which is adjacent to both w and z , then the path $(x = x_0, x_1, \dots, x_i = w, a, z = x_j, \dots, x_k = y)$ is an x, y -path not containing any vertex of $N[b]$; thus (x, y) cannot be a DP, a contradiction.

Otherwise, w.l.o.g. assume that a is the unique vertex of $\{a, b, c\}$ adjacent to w and that c is the unique vertex of $\{a, b, c\}$ adjacent to z . Since $\{a, b, c\}$ is an AT there is an a, c -path Q avoiding $N[b]$. Consequently, there is an x, y -path avoiding $N[b]$ consisting of the x, w -subpath of P followed by the path Q and the z, y -subpath of P . Hence (x, y) is not a DP, a contradiction. \square

Remark 3.1. Of course the necessary condition of Lemma 3.7 for a graph to have a DP is not sufficient. Even if no connected induced subgraph of a graph G has a 3-distant AT, the graph G may not be a weak dominating pair graph. (Consider, e.g., C_7 .)

4. A polar theorem. The following *polar theorem* is one of the main contributions of our paper. It extends the polar theorem for dominating pairs in AT-free graphs of diameter at least 4 given in [3] to weak dominating pair graphs of diameter at least 5.

THEOREM 4.1. Let $G = (V, E)$ be any weak dominating pair graph with diameter at least 5. Then there are disjoint sets $X \subseteq V$ and $Y \subseteq V$ such that, for all $x, y \in V$, (x, y) is a dominating pair of G if and only if $x \in X$ and $y \in Y$.

Proof. Suppose $G = (V, E)$ is a connected graph with $\text{diam}(G) \geq 5$, and let (x, y) be a DP of G . $\text{diam}(G) \leq d_G(a, b) + 2$ for any DP (a, b) of G ; thus $d_G(a, b) \geq 3$ and, in particular, $d_G(x, y) \geq 3$. Let S be any minimal x, y -separator of G . By

Lemma 3.5 only the components of $G[V \setminus S]$ that contain x and y , called C_x and C_y , can be far components of $G[V \setminus S]$.

For a vertex $v \in V \setminus S$ we define $depth(v) = \min_{s \in S} d_G(v, s)$, and we set $depth(s) = 0$ for all vertices $s \in S$. Now we partition the vertex set $S \cup C$ for any component C of $G[V \setminus S]$ into levels $H_k(C) = \{v \in C : depth(v) = k\}$, $k \geq 0$, containing all vertices of depth k . We say that the maximum k for which $H_k(C) \neq \emptyset$ is the depth of the component C and denote it by $depth(C)$. Thus $depth(C) = 1$ for all nonfar components of $G[V \setminus S]$.

Let (a, b) be any DP of G . Since $diam(G) \geq 5$ and since every a, b -path is dominating, one of the vertices a and b must belong to the last two levels of C_x and the other one to the last two levels of C_y .

Now, for the DP (x, y) of G , we define the sets $X = X(x, y)$ and $Y = Y(x, y)$ as follows. If (a, b) is a DP, then $a \in X$ and $b \in Y$ if and only if either $a \in S$ and $\{a, x\} \in E$ or $a \in C_x$. Notice that the partition is well defined, since x and y have no common neighbor. Notice that the theorem is true if, for every connected graph G with $diam(G) \geq 5$, (a, b) is a DP of G if and only if $a \in X$ and $b \in Y$.

Suppose there is a connected graph $G = (V, E)$ with diameter at least 5 violating the theorem. Thus there is a DP (x, y) of G and a minimal x, y -separator S , and there exist another DP (x', y') ($x' \neq x$ and $y' \neq y$) of G with $x' \in C_x$, or $x' \in S$ and $\{x', x\} \in E$, such that (x', y') is not a DP of G . As mentioned above, $x' \in H_{depth(C_x)}(C_x) \cup H_{depth(C_x)-1}(C_x)$ and $y' \in H_{depth(C_y)}(C_y) \cup H_{depth(C_y)-1}(C_y)$.

Furthermore, if (a, b) is a DP with $a \in C_x \cup S$ and $b \in C_y \cup S$, then, by Lemma 3.3, every a, S -path can be extended to an a, b -path by adding vertices of C_y and vertex b only, and similarly every b, S -path can be extended to an a, b -path by adding vertices of C_x and vertex a only. Thus since (a, b) is a DP the vertex set of every a, S -path dominates C_x , and the vertex set of every b, S -path dominates C_y . Consequently, the vertex set of every x, S -path as well as the vertex set of every x', S -path dominates C_x , and the vertex set of every y, S -path as well as the vertex set of every y', S -path dominates C_y .

We first establish the following three claims. Then we will complete the proof of the theorem.

Claim 1. There is an x', y -path $P = (x' = x_0, x_1, \dots, x_k = y)$ of G such that $\emptyset \neq V \setminus N[\{x_0, x_1, \dots, x_k\}] \subseteq S$.

Since (x', y) is not a DP there is an x', y -path $P = (x' = x_0, x_1, \dots, x_k = y)$ avoiding $N[z]$ for some vertex z . Clearly, z cannot belong to a nonfar component because this would imply $\{s, z\} \in E$ for all $s \in S$ and every x', y -path has to contain at least one vertex in S .

There is a vertex $\hat{s} \in S$ that belongs to P . Now, the x', \hat{s} -subpath of P as well as the \hat{s}, y -subpath of P avoid $N[z]$. Suppose z belongs to one of the two possible far components, say C_x . Then there is an x', y' -path avoiding $N[z]$ consisting of the x', \hat{s} -subpath of P , a neighbor t of \hat{s} in C_y , and a t, y' -path inside the component C_y . This would imply that (x', y') is not a DP, a contradiction. •

Claim 2. The components C_x and C_y have depth at most 2.

By contradiction and w.l.o.g. we assume $depth(C_x) \geq 3$. Consequently, $depth(x) \geq 2$ and $depth(x') \geq 2$. By Claim 1, there is an x', y -path P and a vertex $z \in S$ such that P avoids $N[z]$. Therefore $N[z] \subseteq S \cup H_1(C_x) \cup H_1(C_y)$, implying x and z are not adjacent. The vertex set of P dominates C_x ; thus the vertex x has a neighbor w in P . Consequently, the path consisting of x and the w, y -subpath of P is an x, y -path avoiding $N[z]$, contradicting the assumption that (x, y) is a DP. •

Claim 3. $d_G(x, y') \leq 2$, and thus (x, y') is not a DP.

By Claim 1, there is an x', y' -path $P = (x' = x_0, x_1, \dots, x_k = y)$ with $\emptyset \neq V \setminus N[\{x_0, x_1, \dots, x_k\}] \subseteq S$. Let z be any vertex of $V \setminus N[\{x_0, x_1, \dots, x_k\}]$; i.e., the path P avoids $N[z]$. Now $x \notin S$ implies that x has a neighbor $w = x_i$ in P . (x, y) is a DP; thus the x, y -path $(x, w = x_i, x_{i+1}, \dots, x_k = y)$ contains a neighbor of z which implies $\{z, x\} \in E$.

If $y' \in V \setminus N[\{x_0, x_1, \dots, x_k\}]$, then $\{x, y'\} \in E$, since $(x, w = x_i, x_{i+1}, \dots, x_k = y)$ contains a neighbor of y' . Otherwise, y' has a neighbor $q = x_j$ in P . Since (x', y') is a DP, the x', y' -path $(x' = x_0, x_1, \dots, x_j = q, y')$ cannot avoid $N[z]$. Thus $\{z, y'\} \in E$ which implies $d_G(x, y') \leq 2$. Consequently, (x, y') cannot be a DP since this would imply $\text{diam}(G) \leq 4$. •

Analogously, starting with the assumption that (x, y) and (x', y') are DPs and that (x, y') is not a DP, we obtain $d_G(x', y) \leq 2$ and that (x', y) is not a DP. Furthermore, $d_G(x, y) \geq 3$ and $d_G(x', y') \geq 3$, since (x, y) and (x', y') are DPs in a graph of diameter at least 5.

Now we are ready to complete the proof of our theorem.

Case 1. $\{x, y'\} \in E$.

Hence $y' \in S$. Notice that $x' \notin S$; otherwise, the definition of X implies $\{x, x'\} \in E$, and thus $d_G(x, y) \leq 2$, a contradiction. Therefore $x' \in C_x$, and, since (x', y') is a DP, y' is adjacent to all vertices in C_y . Therefore y and y' are adjacent, implying $d_G(x, y) \leq 2$, a contradiction.

Case 2. $d_G(x, y') = 2$ and $y' \in S$.

Since $y' \in S$ there is an x', y' -path in which all vertices except y' belong to C_x . Since (x', y') is a DP we obtain that y' is adjacent to all vertices of C_y .

Let (x, t, y') be a shortest x, y' -path. Since y' is adjacent to all vertices in C_y the path $Q = (x, t, y', y)$ is an x, y -path of length 3 with $N[\{x, t, y', y\}] = V$. Therefore $\text{diam}(G) \leq 5$.

Since $\text{diam}(G) \geq 5$ there are vertices w and z with $d_G(w, z) = 5$. W.l.o.g. we may assume that w has only the neighbor x in Q and that z has only the neighbor y in Q .

The vertex y' is adjacent to all vertices in C_y , implying $z \in S$. Furthermore, x' has a neighbor in Q ; otherwise, (x, y) would not be a DP. Since $d_G(x', y') \geq 3$, the only possible neighbor of x' in Q is x ; thus (x', x, t, y') is an x', y' -path and therefore dominating. Consequently, $\{x', z\} \in E$ which implies $d_G(w, z) \leq 3$, contradicting the choice of w and z .

Case 3. $d_G(x, y') = 2$ and $y' \in C_y$.

Let (x, t, y') be a shortest x, y' -path; thus $t \in S$. Since (x', y') is a DP, y is adjacent to t or to y' . Moreover, $\{y, t\} \in E$ would imply $d_G(x, y) \leq 2$, a contradiction. Therefore $\{y, y'\} \in E$. Thus $Q = (x, t, y', y)$ is an x, y -path of length 3 with $N[\{x, t, y', y\}] = V$. Therefore $\text{diam}(G) \leq 5$. As in Case 2, $\text{diam}(G) \geq 5$ implies the existence of vertices w and z with $d_G(w, z) = 5$. We may assume that w has only the neighbor x in Q and z has only the neighbor y in Q .

First let $z \in C_y$. Since (y', t) is a y', S -path, it dominates C_y . Hence z is adjacent to t or y' which contradicts the choice of z . Now let $z \in S$. As in Case 2, x' has a neighbor in Q . Since $d_G(x', y') \geq 3$, the only possible neighbor is x ; thus (x', x, t, y') is an x', y' -path and therefore dominating. Consequently, $\{x', z\} \in E$ which implies $d_G(w, z) \leq 3$, contradicting the choice of w and z .

Thus we have shown that for any weak dominating pair graph G with diameter at least 5, if (x, y) and (x', y') are DPs with $x' \in C_x$ or $x' \in S \cap N(x)$, then (x', y)

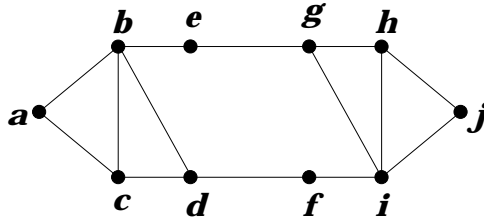


FIG. 4.1. A weak dominating pair graph of diameter 5 without a diametral DP.

and (x, y') are also DPs. This proves the theorem. \square

Remark 4.1. Clearly our polar theorem can be formulated for all graphs with diameter at least 5. However, such a statement will be of no interest for graphs without a DP.

Remark 4.2. The original polar theorem for DPs in AT-free graphs establishes the polar property under the condition that the diameter of the AT-free graph is at least 4. For weak dominating pair graphs the diameter has to be at least 5 to guarantee the polar property, since there is a graph with diameter 4 violating the polar property (see Figure 2.1). Notice that this graph is even a dominating pair graph and has the following five dominating pairs: $(a, g), (b, g), (c, f), (c, h), (c, g)$.

Remark 4.3. One could ask the question, If G is a weak dominating pair graph with diameter at least 5, does G have a diametral DP? However, the answer is negative and a counterexample is given in Figure 4.1. Furthermore, it is easy to see that this counterexample can be extended to any diameter greater than 5. For example, replace the edge $\{e, g\}$ by a path $e = e_0, e_1, \dots, e_t, e_{t+1} = g$ ($t \geq 1$), replace the edge $\{d, f\}$ by a path $d = d_0, d_1, \dots, d_t, d_{t+1} = f$, and add edges $\{e_i, d_i\}$ for all $i \in \{1, 2, \dots, t\}$.

In the last two sections we consider dominating pair graphs. It is not hard to see that the established theorems cannot be extended to weak dominating pair graphs.

5. Chordal dominating pair graphs. We need a lemma on chordal graphs from Farber and Jamison [8]. For more information on chordal graphs we refer the reader to [2, 9].

DEFINITION 5.1. A subset $S \subseteq V$ of a graph G is called m -convex if for any pair of vertices $u, v \in S$ each chordless u, v -path is contained in S .

Hence, if two vertices x and y of an m -convex set S can be joined by a path outside S , i.e., all interior vertices of the path do not belong to S , then x and y must be adjacent.

Let $G = (V, E)$ be a graph, let v be a vertex of G , and let k be a positive integer. The *disk* of radius k centered at v is the set $D_k(v) = \{w \in V : d_G(v, w) \leq k\}$.

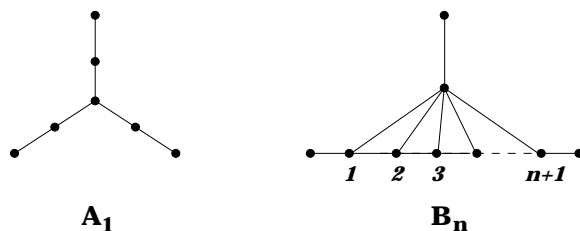
LEMMA 5.2 (see [8]). Any disk $D_k(v)$ of a chordal graph is m -convex.

The next theorem gives a characterization of chordal dominating pair graphs by forbidden induced subgraphs.

THEOREM 5.3. A chordal graph G is a dominating pair graph if and only if it does not contain the graphs A_1 and B_n ($n \geq 1$) as an induced subgraph (see Figure 5.1).

Proof. By Lemma 3.7, the graphs A_1 and B_n , for all $n \geq 1$, do not have a DP, since each of them has a 3-distant AT. Hence a dominating pair graph cannot contain any of the graphs A_1 and B_n , $n \geq 1$, as an induced subgraph.

Suppose the other direction of the theorem would not be true. Let $G = (V, E)$ be

FIG. 5.1. *Forbidden chordal dominating pair graphs.*

a counterexample with the smallest number of vertices. That is, G is a chordal graph that does not contain A_1 and B_n , for each $n \geq 1$, as an induced subgraph, and G is not a dominating pair graph; however, each proper induced connected subgraph of G is a dominating pair graph. Hence G itself is connected and has no DP.

Case 1. $\text{diam}(G) \leq 3$.

We claim that any chordal graph with diameter at most 3 and no B_1 as an induced subgraph has a dominating set of cardinality at most two, which implies that it has a DP.

First, every chordal graph G with $\text{diam}(G) \leq 3$ has a dominating clique [10]. Let C be a minimal dominating clique of G . Thus any vertex of C has a private neighbor with respect to C . On the other hand, since G is chordal and does not contain B_1 as an induced subgraph, there cannot be three different vertices $x, y, z \in C$ such that $\text{Pr}(x, C) \neq \emptyset$, $\text{Pr}(y, C) \neq \emptyset$, and $\text{Pr}(z, C) \neq \emptyset$. Consequently, $|C| \leq 2$. Hence G either has a dominating vertex u or a dominating edge $\{v, w\}$, which implies that either (u, u) or (v, w) is a DP of G .

Therefore the theorem is true for graphs of diameter at most 3.

Case 2. $\text{diam}(G) \geq 4$.

Let (a, b) be a diametral pair of G , i.e., $d_G(a, b) = \text{diam}(G) \geq 4$. Let $P = (a = a_0, a_1, a_2, \dots, a_r = b)$ be a shortest a, b -path in G . Let $u = a_2$. Clearly neither a nor b belongs to $N_G[u]$.

We claim that a and b are in different components of $G[V \setminus N_G[u]]$. We prove the claim by contradiction. Let \tilde{P} be an a, b -path in $G[V \setminus N_G[u]]$. Then concatenating the paths (a_1, a) , \tilde{P} , and $(b = a_r, a_{r-1}, \dots, a_3)$ we obtain a path between a_1 and a_3 outside $N_G[u]$. By Lemma 5.2, $D_1(u) = N_G[u]$ is m -convex and thus a_1 and a_3 are adjacent, contradicting the choice of P . Hence a and b are in different components of $G[V \setminus N_G[u]]$.

Recall that the graph $G - u$ is a chordal dominating pair graph by the choice of G . We distinguish two cases.

Subcase A. $G - u$ is disconnected.

First assume that a and b belong to the same component of $G - u$. Thus there is a shortest a, b -path $Q = (a = b_0, b_1, \dots, b_s = b)$ in $G - u$. Hence Q is an a, b -path in G not containing u . If no vertex of Q is adjacent to u , then $(a_3, a_4, \dots, a_r = b = b_s, b_{s-1}, \dots, b_1, b_0 = a, a_1)$ is an a_1, a_3 -path outside $N_G[u]$. Thus the chordality of G and Lemma 5.2 imply that a_1 and a_3 are adjacent, a contradiction to the choice of P .

Consequently, there is at least one vertex of Q that is a neighbor of u . By the chordality of G , and since $\{a, u\} \notin E$ and $\{b, u\} \notin E$, we obtain that $N_G[u] \cap V(Q) = \{b_i, b_{i+1}, \dots, b_j\}$ for suitable i, j with $1 \leq i \leq j \leq s - 1$. Let v be a neighbor of u in a component of $G - u$ different from the one containing a and b . Then

$\{u, v, b_{i-1}, b_i, b_{i+1}, \dots, b_j, b_{j+1}\}$ induces an A_1 in G , if $i = j \notin \{1, s - 1\}$ and it induces a B_{j-i} in G , if $i < j$. Hence the remaining subcase is $i = j \in \{1, s - 1\}$.

Assume $i = j = s - 1$. Then $(a_1, a, b_1, b_2, \dots, b_{s-1})$ is a path in G between two neighbors of u , and this path is outside of $D_1(u) = N_G[u]$. Therefore Lemma 5.2 implies $\{a_1, b_{s-1}\} \in E$ and thus $d_G(a, b) \leq 3$, contradicting the choice of (a, b) . Assume $i = j = 1$. Then $(a_3, a_4, \dots, a_{r-1}, b, b_{s-1}, \dots, b_1)$ is a path in G between two neighbors of u , and it is outside of $N_G[u]$. Thus Lemma 5.2 implies $\{a_3, b_1\} \in E$. Hence $d_G(a, b) \leq r - 1 = \text{diam}(G) - 1$, contradicting the choice of (a, b) .

Now assume that a and b are in different components of $G - u$. Hence $G - u$ has at least two far components, namely, C_a and C_b . Notice that there are at most two far components of $G - u$, since otherwise A_1 is an induced subgraph of G , contradicting the choice of G . Clearly, these far components are C_a and C_b .

Consider any far component C of $G - u$. Let v_C be a neighbor of u in any other component of $G - u$. Then the graph $G[C \cup \{u, v_C\}]$ is a connected proper induced subgraph of G and has a DP (x, y) . By the construction of $G[C \cup \{u, v_C\}]$, we may assume $x = x_C \in C$ and $y = u$. This establishes the existence of a DP (x_{C_a}, u) of $G[C_a \cup \{u, v_{C_a}\}]$ and a DP (x_{C_b}, u) of $G[C_b \cup \{u, v_{C_b}\}]$. Consequently, (x_{C_a}, x_{C_b}) is a DP of G , since every x_{C_a}, x_{C_b} -path in G passes through u . This contradicts the choice of G as a graph without a DP.

Subcase B. $G - u$ is connected.

Thus $G - u$ has a DP. We choose a DP (x, y) of $G - u$ such that $d_G(x, y)$ is as small as possible. By the choice of G , (x, y) is not a DP of G . Hence there is a vertex $w \in V \setminus (N_G[x] \cup N_G[y])$ of G such that x and y belong to one component of $G[V \setminus N_G[w]]$. Let P' be a chordless x, y -path in G avoiding $N_G[w]$.

We claim that x and y belong to different components of $G[V \setminus N_G[u]]$. To prove the claim, we suppose that x and y belong to the same component C of $G[V \setminus N_G[u]]$. By the choice of u , $G[V \setminus N_G[u]]$ is disconnected. Let z be any vertex of a component of $G[V \setminus N_G[u]]$ different from the one containing x and y . Thus any x, y -path inside C is a path in $G - u$ and contains no vertex of $N_{G-u}[z]$, contradicting the choice of (x, y) . Therefore x and y belong to different components of $G[V \setminus N_G[u]]$. Therefore $w \neq u$. For the remainder of the proof we denote by C_x and C_y , respectively, the component of $G[V \setminus N_G[u]]$ containing x and y , respectively.

The x, y -path P' avoids $N_G[w]$ in G . Since every x, y -path in $G - u$ contains a vertex of $N_{G-u}[w]$, P' is not a path in $G - u$ and thus P' contains u . Furthermore, u and w are not adjacent in G .

Let $P' = (x, \dots, u'_x, u_x, u, u_y, u'_y, \dots, y)$. Since $G - u$ is connected, there is a shortest u_x, u_y -path Q' in $G - u$. Since Q' is chordless, $N_G[u]$ is m -convex, and $\{u_x, u_y\} \notin E$, we conclude that all vertices of Q' are neighbors of u .

Consider the x, y -path in $G - u$ obtained by replacing the subpath (u_x, u, u_y) in P' by Q' . This path contains a neighbor of w since (x, y) is a DP in $G - u$, which implies that Q' contains a neighbor of w , say w' . By the chordality of G and since P' is chordless, the neighbors of w' in P' form a subpath $(c_i, c_{i+1}, \dots, c_j)$ of P' containing u . If $i = j$, then $c_i = u$ and $\{u'_x, u_x, u, u_y, u'_y, w', w\}$ induces an A_1 , a contradiction.

Therefore $i < j$. Suppose w' is adjacent neither to x nor to y . Then $\{c_{i-1}, c_i, \dots, c_j, c_{j+1}, w', w\}$ induces a B_{j-i} in G , a contradiction. Otherwise, we may assume $\{x, w'\} \in E$. By our choice of (x, y) , (w', y) is not a DP of $G - u$. Thus there is a vertex z and a w', y -path in $G - u$ avoiding $N_{G-u}[z]$. Since $\{x, w'\} \in E$, we can add the vertex x to this path and obtain a new x, y -path in $G - u$. This new

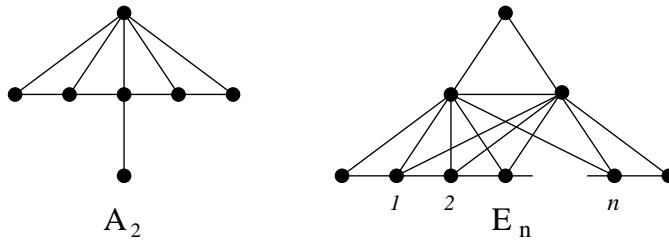


FIG. 5.2. Forbidden interval graphs.

path cannot avoid $N_{G-u}[z]$, implying $\{z, x\} \in E$. z and w are not adjacent in G ; otherwise, (w, w', x, z) is a chordless cycle of length 4, contradicting the chordality of G . Furthermore, z and u are not adjacent; otherwise, (z, x, w', u) is a chordless cycle of length 4, a contradiction. Hence $z \in C_x$. Note that z and u_y are not adjacent in G ; otherwise, $G[\{z, x, \dots, u'_x, u_x, u, u_y\}]$ contains a chordless cycle of length at least 4, contradicting the chordality of G . Consequently, z is not adjacent to any vertex in the subpath (u, u_y, u'_y, \dots, y) of P' . Let c_l be the neighbor of z in P' that is closest to y . Then assuming that w' and y are not adjacent, $\{z, c_l, c_{l+1}, \dots, c_j, c_{j+1}, w', w\}$ induces a B_{j-l} in G , a contradiction.

Finally, suppose that w' is a common neighbor of x and y . By the choice of u , $\text{diam}(G-u) \geq \text{diam}(G) \geq 4$. Thus $d_G(x, y) = 2$ and (x, y) being a DP in $G-u$ imply that either (a, x, w', y, b) or (a, y, w', x, b) is a shortest a, b -path in $G-u$ and in G . W.l.o.g. assume that (a, x, w', y, b) is such a path. By the chordality of G , neither a nor b is adjacent to w . By the choice of u , a and b are not adjacent to u in G . Therefore a belongs to C_x and b belongs to C_y .

Now let $c_{i'}$ be the neighbor of a in the x, u -subpath of P' that is closest to u . Let $c_{j'}$ be the neighbor of b in the u, y -subpath of P' that is closest to u . Hence $j' - i' \geq 2$. Consequently, $\{a, c_{i'}, c_{i'+1}, \dots, c_{j'}, b, w', w\}$ induces a $B_{j'-i'}$ in G , contradicting the choice of G .

Therefore the theorem is true for graphs of diameter at least 4, and this completes the proof. \square

Theorem 5.3 gives a characterization of a certain subclass of chordal graphs by forbidden induced subgraphs.

The best-known theorem of this type was established by Lekkerkerker and Boland in 1962. They showed that interval graphs are exactly the chordal AT-free graphs [12]. They also established the following characterization of interval graphs.

THEOREM 5.4 (see [12]). *A chordal graph G is AT-free if and only if it does not contain the graphs A_1, A_2, B_n ($n \geq 1$), and E_n ($n \geq 1$) as an induced subgraph (see Figures 5.1 and 5.2).*

Furthermore, a characterization of chordal diametral path graphs, showing that A_1 and B_1 are the two minimal forbidden induced subgraphs, is given in [7].

The following corollaries are immediate consequences of Theorem 5.3.

COROLLARY 5.5. *Let $G = (V, E)$ be a chordal graph. Then G is a dominating pair graph if and only if no connected induced subgraph of G has a 3-distant AT.*

COROLLARY 5.6. *A tree is a dominating pair graph if and only if it does not contain the graph A_1 (see Figure 5.1) as an induced subgraph.*

6. PATH-MCDS. We start with a few definitions. A dominating set D of G is called a *connected dominating set* if $G[D]$ is connected. The *connected domination*

number denoted by $\gamma_{\text{conn}}(G)$ is the minimum cardinality of a connected dominating set in G .

DEFINITION 6.1. *A minimum cardinality connected dominating set D (short MCDS) of G is a PATH-MCDS if $G[D]$ is a path on $|D|$ vertices, i.e., $G[D] \cong P_{|D|}$. A graph class \mathfrak{G} has the PATH-MCDS property if every connected graph $G \in \mathfrak{G}$ has a PATH-MCDS.*

Some subclasses of the class of dominating pair graphs are known to have the PATH-MCDS property, among them permutation graphs [5], cocomparability graphs [11], and AT-free graphs [3]. Furthermore, it is known that every connected diametral path graph with diameter at least 5 has a PATH-MCDS [7]. In this section we show that every connected dominating pair graph of diameter not equal to 3 has a PATH-MCDS. We start with two easy lemmas.

LEMMA 6.2. *Let $G = (V, E)$ be a graph with a dominating pair that is not diametral. Then G has a PATH-MCDS.*

Proof. Let (x, y) be a DP of G with $d_G(x, y) < \text{diam}(G)$. Let $P = (x = x_0, x_1, \dots, x_k = y)$ be a shortest x, y -path in G . Clearly, $V(P) = \{x_0, x_1, \dots, x_k\}$ is a connected dominating set of G . Furthermore, $\gamma_{\text{conn}}(G) \geq \text{diam}(G) - 1$, since any MCDS contains the $\text{diam}(G) - 1$ internal vertices of some shortest path between the two vertices of a diametral pair. Consequently, $V(P)$ is a PATH-MCDS if $|V(P)| = \text{diam}(G) - 1$.

Suppose $|V(P)| = \text{diam}(G)$. Then either $\gamma_{\text{conn}}(G) = |V(P)|$ and $V(P)$ is a PATH-MCDS, or $\gamma_{\text{conn}}(G) = |V(P)| - 1 = \text{diam}(G) - 1$. In the latter case G has a PATH-MCDS since any MCDS D contains the $\text{diam}(G) - 1$ internal vertices of a shortest path between the two vertices of a diametral pair. With $\gamma_{\text{conn}}(G) = \text{diam}(G) - 1$ we obtain that D is exactly the set of those vertices. \square

LEMMA 6.3. *Let $G = (V, E)$ be a graph with a diametral DP (x, y) . Then the following implications hold:*

1. *If $\gamma_{\text{conn}}(G) \neq \text{diam}(G)$, then G has a PATH-MCDS.*
2. *If $\gamma_{\text{conn}}(G) = \text{diam}(G)$ and D is an MCDS of G such that $G[D]$ is not a path, then $x, y \notin D$, and $d_{G[D]}(a, b) = \text{diam}(G) - 2$ for all $a \in N(x) \cap D$ and $b \in N(y) \cap D$.*

Proof. Let (x, y) be a diametral DP of G . Let $P = (x = x_0, x_1, \dots, x_k = y)$ be a shortest x, y -path in G . Hence $k = d_G(x, y) = \text{diam}(G)$, and P is dominating. Then $\gamma_{\text{conn}}(G) \leq k + 1$, since $V(P)$ is a connected dominating set. Let D be an MCDS of G . Then $k - 1 \leq |D| = \gamma_{\text{conn}}(G) \leq k + 1$, since $\gamma_{\text{conn}}(G) \geq \text{diam}(G) - 1$. Let $a \in D \cap N[x]$ and $b \in D \cap N[y]$. If $\gamma_{\text{conn}}(G) = k + 1$, then P is a PATH-MCDS. If $\gamma_{\text{conn}}(G) = k - 1$, then D is a PATH-MCDS, since otherwise $d_{G[D]}(a, b) < k - 2$ implies $d_G(x, y) < k$, a contradiction.

Finally, suppose $\gamma_{\text{conn}}(G) = |D| = k$ and $G[D]$ is not a path. Let $a \in N[x] \cap D$ and $b \in N[y] \cap D$. Clearly, $d_{G[D]}(a, b) \leq k - 2$. As above $d_{G[D]}(a, b) < k - 2$ contradicts the choice of (x, y) . Hence, $d_{G[D]}(a, b) = k - 2$ and $x, y \notin D$. \square

We decompose our proof that every connected dominating pair graph of diameter not equal to 3 has a PATH-MCDS into two cases that will be given in Propositions 6.4 and 6.5.

PROPOSITION 6.4. *Every dominating pair graph G with $\text{diam}(G) \in \{1, 2, 4\}$ has a PATH-MCDS.*

Proof. Suppose $G = (V, E)$ is a dominating pair graph with $\text{diam}(G) \in \{1, 2, 4\}$ that has no PATH-MCDS. Then $\text{diam}(G) = \gamma_{\text{conn}}(G)$ by Lemmas 6.2 and 6.3. Since any MCDS of cardinality at most 2 is a PATH-MCDS we obtain that each dominating

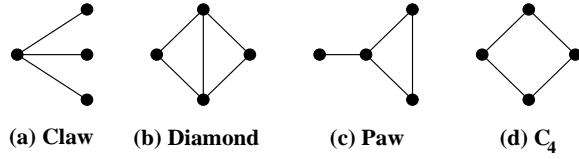


FIG. 6.1.

pair graph G with $\text{diam}(G) \leq 2$ has a PATH-MCDS.

Thus, for our proof by contradiction, we may assume that $G = (V, E)$ is a dominating pair graph with $\text{diam}(G) = 4$ that has no PATH-MCDS. Let (x, y) be a DP of G . By Lemma 6.2, (x, y) is a diametral DP of G . Thus $d_G(x, y) = 4$ for every DP (x, y) . By Lemma 6.3, $\gamma_{\text{conn}}(G) = 4$ and $\text{diam}(G[D]) \geq 2$ for any MCDS D of G . Thus any MCDS D of G induces one of the four connected graphs on four vertices, depicted in Figure 6.1.

Let (x, y) be a DP of G . Throughout the proof we assume $D = \{a, a_1, b, u\}$, where $a \in D \cap N(x)$, $b \in D \cap N(y)$, and (x, a, a_1, b, y) is a shortest x, y -path in G and thus dominating. Since G has no PATH-MCDS, no vertex set D with $G[D] \cong P_4$ is dominating; i.e., G has no dominating P_4 . Consequently, neither (x, a, a_1, b) nor (a, a_1, b, y) is dominating, which implies that $Pr(u, D)$ can be partitioned into the two nonempty sets $U_x = N(x) \cap Pr(u, D)$ and $U_y = N(y) \cap Pr(u, D)$. Notice that $\{u_x, u_y\} \notin E$ for all $u_x \in U_x$ and all $u_y \in U_y$; otherwise, $d_G(x, y) \leq 3$, a contradiction.

Case 1. $G[D]$ is isomorphic to a claw, i.e., $N(u) \cap \{a, a_1, b\} = \{a_1\}$.

First, $d_G(x, y) = 4$ implies $N[x] \cap N[y] = \emptyset$. Hence either $u \notin N[\{x, y\}]$ or u is adjacent to exactly one of x and y ; w.l.o.g. we may assume $\{u, x\} \notin E$.

Let $u_x \in U_x$ and $u_y \in U_y$. Consider the path (a, a_1, u, u_y) that cannot be dominating. Thus there is a vertex $w \in Pr(b, D \cup \{u_y\})$. Furthermore, $\{w, x\} \notin E$, since $\{w, x\} \in E$ would imply $d_G(x, y) \leq 3$, a contradiction. Altogether we obtain that $\{x, a, a_1, b, w, u, u_y\}$ induces an A_1 (irrespective of whether $\{u, y\} \in E$ or not), which contradicts the choice of G as a dominating pair graph. Thus no counterexample G can have an MCDS D for which $G[D]$ is isomorphic to a claw.

Case 2. $G[D]$ is isomorphic to a diamond, i.e., $N(u) \cap \{a, a_1, b\} = \{a, a_1, b\}$.

The vertex u is neither adjacent to x nor to y ; otherwise, $d_G(x, y) \leq 3$, a contradiction. Let $u_x \in U_x$ and $u_y \in U_y$. (x, y) is a DP of G ; thus (x, u_x, u, u_y, y) is a shortest x, y -path in G and is dominating. On the other hand, neither (x, u_x, u, u_y) nor (u_x, u, u_y, y) is dominating, since G has no dominating P_4 . Hence there is a vertex $w_x \in N(x) \setminus N(y)$ with $w_x \in N[\{a, a_1, b\}] \setminus N[\{u_x, u, u_y\}]$, and there is a vertex $w_y \in N(y) \setminus N(x)$ with $w_y \in N[\{a, a_1, b\}] \setminus N[\{u_x, u, u_y\}]$.

The x, y -path (x, a, u, b, y) is dominating, since (x, y) is a DP. Furthermore, neither (x, a, u, b) nor (a, u, b, y) is dominating. Hence $Pr(a_1, D)$ can be partitioned into the two nonempty sets $A_1^x = N(x) \cap Pr(a_1, D)$ and $A_1^y = N(y) \cap Pr(a_1, D)$. Let $a_1^x \in A_1^x$ and $a_1^y \in A_1^y$. Then $\{a_1^x, y\} \notin E$ implies $\{a_1^x, u_x\} \in E$, since $\{a_1^x, y, u_x, a_1, b, u\}$ cannot induce a B_1 in the dominating pair graph G . Similarly, $\{a_1^y, x\} \notin E$ implies $\{a_1^y, u_y\} \in E$, since $\{a_1^y, x, u_y, a, a_1, u\}$ cannot induce a B_1 in G .

Therefore any vertex of $Pr(a_1, D)$ is adjacent to u_x or to u_y . By our assumptions in the first paragraph of this case, $w_x, w_y \notin N[\{u_x, u, u_y\}]$. It follows that neither w_x nor w_y belongs to $Pr(a_1, D)$. Furthermore, $d_G(x, y) = 4$ implies $\{a, w_y\} \notin E$,

$\{b, w_x\} \notin E$, and $\{w_x, w_y\} \notin E$. If $w_x \in Pr(a, D)$ and $w_y \in Pr(b, D)$, then $\{w_x, u_x, w_y, a, a_1, b, u\}$ induces a B_2 , contradicting the fact that G is a dominating pair graph. Otherwise, either $N(w_x) \cap D = \{a, a_1\}$ and $N(w_y) \cap D = \{b\}$, or $N(w_x) \cap D = \{a\}$ and $N(w_y) \cap D = \{a_1, b\}$. Then either $\{w_x, w_y, u_x, a_1, b, u\}$ or $\{w_x, w_y, u_x, a, a_1, u\}$ induces a B_1 in G , a contradiction.

Thus no counterexample G can have an MCDS D for which $G[D]$ is isomorphic to a diamond.

Case 3. $G[D]$ is isomorphic to a paw; w.l.o.g. $N(u) \cap \{a, a_1, b\} = \{a, a_1\}$.

Then $d_G(x, y) = 4$ implies $y \in Pr(b, D)$, while either $x \in Pr(a, D)$ or $x \in (N[a] \cap N[u]) \setminus N[\{a_1, b\}]$. Thus $x \notin Pr(a, D)$; otherwise, $\{x, u_y, b, a, a_1, u\}$ induces a B_1 in G .

Therefore $x \in (N[a] \cap N[u]) \setminus N[\{a_1, b\}]$. The path (u_x, u, a_1, b) is not dominating, since G has no dominating P_4 . Therefore there is a $w \in Pr(a, D \cup \{u_x\})$. Hence $\{w, u_x, b, a, a_1, u\}$ induces a B_1 in the dominating pair graph G , a contradiction. Thus no counterexample G can have an MCDS D for which $G[D]$ is isomorphic to a paw.

Case 4. $G[D]$ is isomorphic to a C_4 , i.e., $N(u) \cap \{a, a_1, b\} = \{a, b\}$.

Then $d_G(x, y) = 4$ implies $x \in Pr(a, D)$ and $y \in Pr(b, D)$. Now the x, y -paths (x, a, a_1, b, y) and (x, a, u, b, y) are dominating in G . Therefore $Pr(a_1, D \cup \{x, y\}) \subseteq N[u]$.

We claim that either (x, a, u, b) or (a, u, b, y) is dominating. To see this we consider the sets $N[x] \cap N[a_1]$ and $N[y] \cap N[a_1]$. If $N[x] \cap N[a_1] = \{a\}$, then (a, u, b, y) is dominating, and, if $N[y] \cap N[a_1] = \{b\}$, then (x, a, u, b) is dominating. Thus we may assume $N[x] \cap N[a_1] \neq \{a\}$ and $N[y] \cap N[a_1] \neq \{b\}$. For every $s \in N[x] \cap N[a_1]$ and for every $t \in N[y] \cap N[a_1]$, the x, y -path (x, s, a_1, t, y) is dominating, which implies $u \in N[s] \cup N[t]$. Consequently, either $N[u] \supseteq (N[x] \cap N[a_1]) \setminus N[\{a, b\}]$ or $N[u] \supseteq (N[y] \cap N[a_1]) \setminus N[\{a, b\}]$. Thus either (a, u, b, y) is dominating or (x, a, u, b) is dominating, a contradiction. Hence no counterexample G can have an MCDS D for which $G[D]$ is isomorphic to a C_4 .

This completes the proof. \square

PROPOSITION 6.5. *Every connected dominating pair graph with diameter at least 5 has a PATH-MCDS.*

Proof. Suppose $G = (V, E)$ is a dominating pair graph with $diam(G) = k > 4$ that has no PATH-MCDS. Let (x, y) be a DP of G . By Lemma 6.2 (x, y) must be a diametral DP of G . Furthermore, $\gamma_{conn}(G) = diam(G)$ by Lemma 6.3.

Let D be an MCDS of G that does not induce a path. By Lemma 6.3, there is an a, b -path in $G[D]$ between a vertex $a \in N(x) \cap D$ and a vertex $b \in N(y) \cap D$ such that $d_{G[D]}(a, b) = k - 2$. Let $A = (a = a_1, a_2, \dots, a_{k-1} = b)$ be such a path. Let u be the only vertex of D not belonging to the path A .

The path $A' = (x = a_0, a = a_1, a_2, \dots, a_{k-1} = b, a_k = y)$ is dominating, since (x, y) is a DP in G . If any of the two paths $(x = a_0, a = a_1, a_2, \dots, a_{k-1} = b)$ and $(a = a_1, a_2, \dots, a_{k-1} = b, a_k = y)$ is dominating, then it is a PATH-MCDS of G . Hence we may assume that this is not the case. Therefore there is a vertex $u_x \in Pr(x, V(A'))$ and a vertex $u_y \in Pr(y, V(A'))$. Clearly, both are neighbors of u . Hence (x, u_x, u, u_y, y) is a path in G , which is a contradiction to the assumption $d_G(x, y) = diam(G) > 4$. \square

The following theorem summarizes Propositions 6.4 and 6.5.

THEOREM 6.6. *Every connected dominating pair graph of diameter not equal to 3 has a PATH-MCDS.*

7. Open problems. It is possible that other results known for AT-free graphs can be generalized to dominating pair graphs. For example, it is known that the strong perfect graph conjecture is true for AT-free graphs (see [3]), and it is a challenging problem to show that the strong perfect graph conjecture is true for dominating pair graphs.

Despite our strong efforts we were not able to settle the question of whether every dominating pair graph of diameter 3 has a PATH-MCDS. Finally, it is a very interesting open question whether there is a polynomial time algorithm to recognize dominating pair graphs.

Acknowledgment. We are grateful to the referees for their thorough reading and many helpful comments.

REFERENCES

- [1] J.A. BONDY AND U.S.R. MURTY, *Graph Theory with Applications*, American Elsevier, New York, 1976.
- [2] A. BRANDSTÄDT, V.B. LE, AND J.P. SPINRAD, *Graph Classes: A Survey*, SIAM Monogr. Discrete Math. 3, SIAM, Philadelphia, 1999.
- [3] D.G. CORNEIL, S. OLARIU, AND L. STEWART, *Asteroidal triple-free graphs*, SIAM J. Discrete Math., 10 (1997), pp. 399–430.
- [4] D.G. CORNEIL, S. OLARIU, AND L. STEWART, *Linear time algorithms for dominating pairs in asteroidal triple-free graphs*, SIAM J. Comput., 28 (1999), pp. 1284–1297.
- [5] C.J. COLBOURN AND L.K. STEWART, *Permutation graphs: Connected domination and Steiner trees*, Discrete Math., 86 (1991), pp. 179–190.
- [6] J.S. DEOGUN, *Diametral path graphs*, presented at the Second International Conference on Graph Theory and Algorithms, San Francisco, 1989.
- [7] J. DEOGUN AND D. KRATSCH, *Diametral path graphs*, in Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science, Aachen, Germany, 1995, Lecture Notes in Comput. Sci. 1017, Springer-Verlag, Berlin, 1995, pp. 344–357.
- [8] M. FARBER AND R.E. JAMISON, *Convexity in graphs and hypergraphs*, SIAM J. Algebraic Discrete Math., 7 (1986), pp. 433–444.
- [9] M.C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [10] D. KRATSCH, P. DAMASCHKE, AND A. LUBIW, *Dominating cliques in chordal graphs*, Discrete Math., 128 (1994), pp. 269–276.
- [11] D. KRATSCH AND L. STEWART, *Domination on cocomparability graphs*, SIAM J. Discrete Math., 6 (1993), pp. 400–417.
- [12] C.G. LEKKERKERKER AND J.CH. BOLAND, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

MULTIDIMENSIONAL CONVOLUTIONAL CODES*

BRUCE KITCHENS†

Abstract. Several formulations for the definition of a multidimensional convolutional code are discussed in a symbolic dynamics setting and then one is adopted. Using this definition it is shown that there are five other equivalent formulations. One is in terms of the dual space of linear functionals, one is in terms of a parity check system, one is in terms of an algebraic conjugacy, another is in terms of a convolutional encoder, and the last is in terms of a convolutional decoder. Finally, two examples of two-dimensional convolutional codes are discussed in detail.

Key words. convolutional codes, algebraic codes, multidimensional codes

AMS subject classifications. 94B10, 94B05, 94A08

PII. S0895480100378495

1. Introduction. The purpose of this paper is to find a suitable formulation for the definition of a multidimensional convolutional code from a symbolic dynamics perspective. The study of symbolic dynamical systems and the study of modulation coding has had a mutually profitable interaction in the past; see, for example, [11] or [1]. In the past 15 years there has been a rapid development in the theory of multidimensional algebraic dynamical systems; see, for example, [14]. It seems reasonable to hope that the study of multidimensional convolutional coding and the study of multidimensional algebraic dynamical systems will also have a useful interaction.

There is a considerable difference between a one-dimensional convolutional code and a two-dimensional code. The difference mirrors the fact that a polynomial ring in one variable with coefficients in a field is a principal ideal domain, while a polynomial ring in more than one variable with coefficients in a field is not a principal ideal domain. We examine three possible formulations for a definition of convolutional code. They are in terms of the space of linear functionals for the code. The three definitions agree in one dimension, and we illustrate with examples that they differ in more than one dimension. After adopting one of the formulations as a definition we show there are five conditions which are equivalent to the definition. One of the conditions is in terms of the dual space of linear functionals, one is in terms of a parity check system for the code, one is in terms of an algebraic conjugacy with another symbolic dynamical system, another is in terms of the existence of a convolutional encoder, and the last is in terms of the existence of a convolutional decoder. We give two examples of two-dimensional convolutional codes and then analyze them in terms of the five equivalent conditions.

The paper is organized as follows. Section 2 contains background material on symbolic dynamical systems, the algebra of polynomial rings, modules over these rings, and finally how these ideas are related using linear functionals. In section 3 we state three possible formulations for the definition of a convolutional code. Each successive definition implies the preceding one. Then using the fundamental structure theorem for modules over a principal ideal domain we see that the three conditions are equivalent for one-dimensional codes. Following that there are three examples which

*Received by the editors September 20, 2000; accepted for publication (in revised form) March 18, 2002; published electronically May 23, 2002.

<http://www.siam.org/journals/sidma/15-3/37849.html>

†IBM Research, Yorktown Heights, NY 10598 (brucek@us.ibm.com).

show that in two dimensions the conditions are all different. In section 4 there are five conditions on codes which are shown to be equivalent for a code to be a convolutional code. Finally, in section 5 there are two examples of two-dimensional codes which are analyzed in terms of the definition of a convolutional code and the five equivalent conditions.

2. Background. First we will cover some background material on symbolic dynamical systems and then review some algebra of polynomial rings and of modules over polynomial rings. After that we will see how these ideas are related through duality. Both [4] and [8] contain introductions to symbolic dynamical systems. The algebraic material can be found in [3] or [7]. A general introduction to the type of algebraic dynamical systems we will be dealing with can be found in [14], and a more detailed study can be found in [13]. A very nice introduction to one-dimensional convolutional codes is in [12]. Connections between symbolic dynamical systems and one-dimensional convolutional codes can be found in [5], and discussions of multidimensional convolutional codes from slightly different perspectives can be found in [2] and in [15].

The spaces used in symbolic dynamics are composed of infinite sequences or arrays over a finite alphabet. We start with a finite symbol set $\{1, \dots, n\}$ which is usually called the *alphabet*. Then form the space of all two-sided infinite sequences on this alphabet. It is denoted by $\{1, \dots, n\}^{\mathbb{Z}}$. A point x in this space is a two-sided infinite sequence of symbols from the alphabet, $x = \dots x_{-2}x_{-1}.x_0x_1x_2\dots$, where a decimal point is usually used to denote the 0th coordinate. So, for each $i \in \mathbb{Z}$, $x_i \in \{1, \dots, n\}$. For each $d = 1, 2, \dots$ we form the space of all d -dimensional arrays on this alphabet. The d -dimensional array space is denoted by $\{1, \dots, n\}^{\mathbb{Z}^d}$. A point $x \in \{1, \dots, n\}^{\mathbb{Z}^d}$ is an infinite d -dimensional array of symbols from the alphabet and for each $(i_1, \dots, i_d) \in \mathbb{Z}^d$, $x_{(i_1, \dots, i_d)} \in \{1, \dots, n\}$. A natural metric can be put on these spaces. It will say that two sequences are close together if they agree for a long time around their center, and two d -dimensional arrays are close together if they agree in a large d -dimensional box around their center. To define the metric on the sequence space we say $d(x, y) = 0$ if $x = y$ and $d(x, y) = 1/2^N$ with N the integer, where $x_i = y_i$ for $|i| < N$ and $x_N \neq y_N$ or $x_{-N} \neq y_{-N}$. To define the metric on the d -dimensional array space we first put the supremum norm on \mathbb{Z}^d which says that $\|(i_1, \dots, i_d)\| = \max\{|i_1|, \dots, |i_d|\}$. Then we define the metric on the d -dimensional array space by saying $d(x, y) = 0$ if $x = y$ and $d(x, y) = 1/2^N$ with N the integer, where $x_{(i_1, \dots, i_d)} = y_{(i_1, \dots, i_d)}$ for every (i_1, \dots, i_d) with $\|(i_1, \dots, i_d)\| < N$ and $x_{(i_1, \dots, i_d)} \neq y_{(i_1, \dots, i_d)}$ for some (i_1, \dots, i_d) with $\|(i_1, \dots, i_d)\| = N$. When $d = 1$ this metric is the same metric as defined for the sequence space. The sequence space, which is the array space with $d = 1$, and the array spaces are compact spaces when this metric is used. When the cardinality of the alphabet is greater than one it can be shown that every array space is homeomorphic to the usual middle thirds Cantor set.

We define the *shift transformation*, which is denoted by σ , on the sequence space. The shift transformation acts on a point x in $\{1, \dots, n\}^{\mathbb{Z}}$ by shifting it to the left by one coordinate; that is, $(\sigma(x))_i = x_{i+1}$ for all $i \in \mathbb{Z}$. In the metric topology we defined, the shift is a continuous map. Its inverse map is the right shift, so σ is a homeomorphism of $\{1, \dots, n\}^{\mathbb{Z}}$ to itself. On the d -dimensional array space we define *d coordinate shift transformations*. For $j = 1, \dots, d$ the j th coordinate shift transformation is denoted by σ_j . It acts on a point x in $\{1, \dots, n\}^{\mathbb{Z}^d}$ by shifting it in the j th coordinate direction by one coordinate; that is, $(\sigma_j(x))_{(i_1, \dots, i_d)} = x_{(i_1, \dots, i_{j-1}, i_j+1, i_{j+1}, \dots, i_d)}$ for all

$(i_1, \dots, i_d) \in \mathbb{Z}^d$. By the same reasoning as for the sequence space each coordinate shift is a homeomorphism. They are commuting homeomorphisms because the shifts along the coordinate directions act independently of each other.

For $d = 1, 2, \dots$ the space $\{1, \dots, n\}^{\mathbb{Z}^d}$ together with its d coordinate shift transformations $\sigma_1, \dots, \sigma_d$ is a d -dimensional dynamical system, and it is called the *full d -dimensional n -shift*. Suppose X is a subset of $\{1, \dots, n\}^{\mathbb{Z}^d}$ which is closed in the metric topology on $\{1, \dots, n\}^{\mathbb{Z}^d}$ and is invariant under each coordinate shift; that is, $\sigma_j(X) = X$. Then X together with its d coordinate shift transformations $\sigma_1, \dots, \sigma_d$ is a d -dimensional dynamical system, and it is called a *d -dimensional subshift*.

Now we make some notational conventions. For any $d = 1, 2, \dots$ we denote by \underline{i} an element of \mathbb{Z}^d , $\underline{i} = (i_1, \dots, i_d) \in \mathbb{Z}^d$. Then on the d -dimensional array space $\{1, \dots, n\}^{\mathbb{Z}^d}$ we denote by $\sigma^{\underline{i}}$ the transformation $\sigma_1^{i_1} \circ \dots \circ \sigma_d^{i_d}$. These conventions will help simplify some of the notation in the equations that follow.

To define a special class of algebraic dynamical systems we start with a finite field \mathbb{F} . The alphabet for the symbolic dynamical systems will be \mathbb{F}^n , the direct sum of \mathbb{F} with itself n times. We form the d -dimensional array space on \mathbb{F}^n which is denoted by $(\mathbb{F}^n)^{\mathbb{Z}^d}$. It has the same metric as defined for an arbitrary alphabet and is again a compact space which is homeomorphic to the usual middle thirds Cantor set when $|\mathbb{F}^n| > 1$. If x and y are two points in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ we can define their sum using coordinate by coordinate addition, $(x + y)_{\underline{i}} = x_{\underline{i}} + y_{\underline{i}}$ for all $\underline{i} \in \mathbb{Z}^d$. If x is a point in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ and α is an element in \mathbb{F} we can multiply x by α using coordinate by coordinate multiplication, $(\alpha x)_{\underline{i}} = \alpha x_{\underline{i}}$ for all $\underline{i} \in \mathbb{Z}^d$. With these operations $(\mathbb{F}^n)^{\mathbb{Z}^d}$ is a vector space over \mathbb{F} , and it is infinite-dimensional.

Suppose σ_j is the j th coordinate shift transformation. If α is an element in \mathbb{F} and x and y are two points in $(\mathbb{F}^n)^{\mathbb{Z}^d}$, then $\sigma_j(x + y) = \sigma_j(x) + \sigma_j(y)$ and $\sigma_j(\alpha x) = \alpha \sigma_j(x)$. This means each σ_j is a vector space automorphism of $(\mathbb{F}^n)^{\mathbb{Z}^d}$. The vector space $(\mathbb{F}^n)^{\mathbb{Z}^d}$ together with the d commuting automorphisms $\sigma_1, \dots, \sigma_d$ is a dynamical system and is called the *full d -dimensional \mathbb{F}^n -shift*. Suppose X is a subshift of $(\mathbb{F}^n)^{\mathbb{Z}^d}$ which is closed under addition, when $x, y \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ then $x + y \in (\mathbb{F}^n)^{\mathbb{Z}^d}$, and multiplication by elements of \mathbb{F} , when $\alpha \in \mathbb{F}$ and $x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ and then $\alpha x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$. When this happens X is a subvector space of $(\mathbb{F}^n)^{\mathbb{Z}^d}$, and we say that it together with the d automorphisms $\sigma_1, \dots, \sigma_d$ is a *d -dimensional vector shift*. In this paper we will be concerned only with vector shifts.

Next we will discuss algebraic maps between vector shifts, the topological entropy or Shannon capacity of a vector shift, some dynamical properties of vector shifts, and some of their relationships. Let \mathbb{F} be any finite field and $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ a vector shift. Define $\mathcal{W}(X, n)$ to be the set of d -dimensional words of size n that can occur in X ,

$$\mathcal{W}(X, n) = \left\{ \left[\begin{array}{ccc} \vdots & & \\ \cdots & x_{\underline{i}} & \cdots \\ \vdots & & \end{array} \right] : x \in X, \underline{i} \text{ with } 0 \leq i_j, \|\underline{i}\| < n \right\}.$$

Let X and Y be two vector shifts contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$. An *algebraic map* φ from X into Y is a continuous vector space homomorphism from X into Y which commutes with all d shift transformations, $\varphi \circ \sigma_j = \sigma_j \circ \varphi$, where the σ_j on the left is the j th coordinate shift on X and the one on the right is the shift on Y . A standard result about symbolic dynamical systems shows that any such map must be a *sliding block*

code. These are known as *block maps* in symbolic dynamics. It means that there is an m and a vector space homomorphism φ' from $\mathcal{W}(X, 2m+1)$ to $\mathcal{W}(Y, 1)$ which defines φ ,

$$(\varphi(x))_{\underline{i}} = \varphi' \left(\begin{bmatrix} \vdots \\ \cdots & x_{\underline{j}} & \cdots \\ \vdots \end{bmatrix} \right),$$

for each $\underline{i} \in \mathbb{Z}^d$ and \underline{j} with $\|\underline{j} - \underline{i}\| \leq m$. If the algebraic map φ takes X onto Y we say that φ is an *algebraic factor map* and that Y is an *algebraic factor* of X . If the homomorphism φ is a homeomorphism between X and Y we say that φ is an *algebraic conjugacy* and that X and Y are *algebraically conjugate*.

To define the topological entropy or Shannon capacity of a vector shift let $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ be a vector shift as above. Then the *topological entropy* usually simply called *entropy* or the *Shannon capacity* of the vector shift X is defined to be

$$h(X) = \lim_{n \rightarrow \infty} \frac{1}{n^d} \log |\mathcal{W}(x, n)|,$$

where $|\mathcal{W}(x, n)|$ denotes the cardinality of the set $\mathcal{W}(x, n)$. Note that there is a strong dependence in this definition on the dimension d of the vector shift. From the coding theory point of view the entropy is a measure of the amount of information which can be stored in the system.

Suppose $X, Y \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ are vector shifts as above. Also from standard results in symbolic dynamics we know that if Y is an algebraic factor of X , then the entropy of Y is less than or equal to the entropy of X and, if X and Y are algebraically conjugate, then their entropies are equal.

There are three dynamical properties of vector shifts that we will use. Let $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ be a vector shift as above. Then X is *transitive* if it contains a point x whose orbit is dense in X ,

$$\overline{\bigcup_{\underline{i} \in \mathbb{Z}^d} \sigma^{\underline{i}}(x)} = X.$$

If $Y \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ is another vector shift which is an algebraic factor of X , then Y is transitive if X is transitive. This follows because the image of a point in X with a dense orbit will have a dense orbit in Y . Another standard result from symbolic dynamics says that if X is a transitive vector shift and Y is a vector shift properly contained in X , then the entropy of Y is strictly less than the entropy of X . Next we say that a transitive vector shift X has *completely positive entropy* if the following condition holds. Whenever $Y \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ is a vector shift with entropy zero which is an algebraic factor of X then Y consists of the single point of all zeros. We will not explicitly use this property, but it is interesting to see how it arises in our discussion. The final dynamical property which we will use is self-explanatory; the property is for a d -dimensional vector shift to be algebraically conjugate to the full d -dimensional \mathbb{F}^k -shift for the proper \mathbb{F} and some k . We will see in section 3 that for vector shifts in dimension one all three conditions are equivalent but in higher dimensions they are not. In dimensions greater than one, if a vector shift is algebraically conjugate to the full d -dimensional \mathbb{F}^k -shift, then it has strictly positive entropy, and if it has

strictly positive entropy, then it is transitive while neither of the converse implications hold. This difference between dimension one and higher dimensions is what makes the theory of vector shifts in dimension one and the theory of vector shifts in higher dimensions quite different.

Next we will review some basic definitions and terminology about rings and modules. The rings which will be of interest are the Laurent polynomial rings in several variables over finite fields. Let \mathbb{F} be finite field and u_1, \dots, u_d be d indeterminates. Then a Laurent polynomial over this field in these indeterminates is a sum of the form

$$\sum_{k=1}^m \alpha_{(i_{1(k)}, \dots, i_{d(k)})} u_1^{i_{1(k)}} \cdots u_d^{i_{d(k)}},$$

where each $(i_{1(k)}, \dots, i_{d(k)}) \in \mathbb{Z}^d$ and each $\alpha_{(i_{1(k)}, \dots, i_{d(k)})} \in \mathbb{F}$. These are the same as polynomials in several variables over a finite field, only now we allow negative as well as positive exponents. We can add or multiply two Laurent polynomials just as we add or multiply two of the usual polynomials. With these operations the set of all Laurent polynomials in the indeterminates u_1, \dots, u_d and coefficients in the field \mathbb{F} form a commutative ring. We will denote this ring by $\mathbb{F}[u_1^{\pm 1}, \dots, u_d^{\pm 1}]$. Just as we did for vector shifts we will make a notational convention; let $\underline{i} \in \mathbb{Z}^d$ and then let $\underline{u}^{\underline{i}}$ stand for the monomial $u_1^{i_1} \cdots u_d^{i_d}$. Let $\mathbb{F}[\underline{u}^{\pm 1}]$ denote the ring $\mathbb{F}[u_1^{\pm 1}, \dots, u_d^{\pm 1}]$. This will help simplify some of the later notation.

Recall that a ring R is an integral domain if it is commutative and has no zero divisors; that is, if $xy = 0 \in R$, then either $x = 0$ or $y = 0$. Also recall the definition of an ideal I in a commutative ring R . If $x_1, \dots, x_\ell \in R$, then the ideal they generate is the set of all their linear combinations $w_1x_1 + \cdots + w_\ell x_\ell$ for any $w_1, \dots, w_\ell \in R$. We denote this ideal by $\langle x_1, \dots, x_\ell \rangle$. A ring is a *principal ideal domain* if it is an integral domain and every ideal can be generated by a single element. The familiar ring of integers \mathbb{Z} is the best known principal ideal domain. An important fact for us is that the ring of Laurent polynomials in a single variable with coefficients in a finite field $\mathbb{F}[u^{\pm 1}]$ is also a principal ideal domain. We need a generalization of principal ideal domain. A ring is a *commutative Noetherian ring* if it is an integral domain and every ideal can be generated by a *finite* number elements. This is crucial for us because the ring of Laurent polynomials in more than one variable over a finite field $\mathbb{F}[\underline{u}^{\pm 1}]$ is a commutative Noetherian ring but not a principal ideal domain. To see this consider the ideal generated by the polynomials $1 + u_1$ and $1 + u_2$ in $\mathbb{F}[u_1^{\pm 1}, u_2^{\pm 1}]$. This ideal cannot be generated by a single element. This follows because the only common divisor of $1 + u_1$ and $1 + u_2$ is 1. This means 1 would have to be the generator, but 1 is not a linear combination of $1 + u_1$ and $1 + u_2$ and so is not in the ideal.

Next recall the definition of a module over a ring. All of the modules we will consider will be modules over the Laurent polynomial rings just discussed $\mathbb{F}[\underline{u}^{\pm 1}]$. Let M be a module over a commutative ring R . If $x_1, \dots, x_\ell \in M$, then the submodule they generate is the set of all their linear combinations $\alpha_1x_1 + \cdots + \alpha_\ell x_\ell$ for any $\alpha_1, \dots, \alpha_\ell \in R$. As for ideals we denote this submodule by $\langle x_1, \dots, x_\ell \rangle$. For the purposes of this paper we choose the term *rank* of a module M to denote the minimum number of elements it takes to generate the entire module. A module M is a *free module* if there is a set of generators that is linearly independent; and no nontrivial linear combination of them is zero. If we let $\mathbb{F}[\underline{u}^{\pm 1}]^k$ denote the direct sum of k copies of $\mathbb{F}[\underline{u}^{\pm 1}]$, then it is a free module over $\mathbb{F}[\underline{u}^{\pm 1}]$ with rank k . A linearly independent set of generators for a free module is a *base* for the module, and every base has the same cardinality. An element x in the module M is a *torsion element* if there is a nonzero

$\alpha \in R$ and $\alpha x = 0$. The set of all torsion elements in M forms a submodule of M . We let $\text{tor}M$ denote this submodule. If a module has no torsion elements, except 0, we say it is *torsion free*. A module may be torsion-free but not be a free module. An example of this is the ideal previously discussed $\langle 1 + u_1, 1 + u_2 \rangle \subseteq \mathbb{F}[u_1^{\pm 1}, u_2^{\pm 1}]$ thought of as a module over $\mathbb{F}[u_1^{\pm 1}, u_2^{\pm 1}]$. However, a finitely generated module over a principal ideal domain is a free module if and only if it is torsion-free. This follows from the fundamental structure theorem for such modules which we will discuss in section 3. So any module over a Laurent polynomial ring in one variable with coefficients in a finite field is free if and only if it is torsion-free. In section 3 we will see an important consequence of this in terms of one- versus higher-dimensional vector shifts.

A module M over a commutative ring R is a *commutative Noetherian module* if the ring R is a commutative Noetherian ring and every submodule of M is finitely generated. The module $\mathbb{F}[\underline{u}^{\pm 1}]^n$ over the ring $\mathbb{F}[\underline{u}^{\pm 1}]$ is a free commutative Noetherian module. A submodule of a commutative Noetherian module is also a commutative Noetherian module. If N is a submodule of the module M , then we can form the *quotient module* M/N . If M is a commutative Noetherian module, then every quotient module of M is also a commutative Noetherian module. In this paper we will deal only with submodules and quotient modules of the modules $\mathbb{F}[\underline{u}^{\pm 1}]^n$ over the rings $\mathbb{F}[\underline{u}^{\pm 1}]$. Every one of these is a commutative Noetherian module, and they are relatively concrete. We will need to use a deep result about modules over polynomial rings. The Quillen–Suslin theorem which is the solution to the Serre conjecture in [6] and [9] states that a finitely generated projective module over a polynomial ring with coefficients in a field is a free module. The version of the theorem that we will use states that if $\mathbb{F}[\underline{u}^{\pm 1}]^n$ can be written as a direct sum of two submodules, $\mathbb{F}[\underline{u}^{\pm 1}]^n = M \oplus N$, then both M and N are free modules.

Next we see how linear functionals for vector spaces and their duality theory are related to the vector shifts and the modules we have just discussed. A *linear functional* for a vector space X over a field \mathbb{F} is a vector space homomorphism from X into \mathbb{F} . The collection of all the linear functionals is itself a vector space using pointwise addition and multiplication of the linear functionals as the operations. The collection of all linear functionals with these operations is the *dual vector space of X* and is denoted \widehat{X} . The dual vector space \widehat{X} of X *separates the points of X* meaning that if $x \neq y \in X$, then there is an $f \in \widehat{X}$ with $f(x) \neq f(y)$. As an example let \mathbb{F} be a finite field and \mathbb{F}^n the vector space. The dual vector space $\widehat{\mathbb{F}^n}$ is isomorphic to \mathbb{F}^n in a natural way. Let $\{e_i\}$ be the standard basis vectors for \mathbb{F}^n . Then a linear functional is determined by the value in \mathbb{F} where it sends each e_i . We denote a vector in the vector space as a row vector and a linear functional by a column vector. The linear functional

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

sends the vector (v_1, \dots, v_n) to $\alpha_1 v_1 + \dots + \alpha_n v_n \in \mathbb{F}$. This natural correspondence between X and \widehat{X} is unusual and happens because the vector space is finite-dimensional. We will see that things are different for vector shifts. Duality theory does show that the dual vector space of \widehat{X} is isomorphic to X in a canonical manner. A point $x \in X$ is a linear functional on \widehat{X} by sending $f \in \widehat{X}$ to $f(x) \in \mathbb{F}$. Now we examine the vector space $(\mathbb{F}^n)^{\mathbb{Z}^d}$. Because of the topology on $(\mathbb{F}^n)^{\mathbb{Z}^d}$ and \mathbb{F} , a linear functional on

the vector space $(\mathbb{F}^n)^{\mathbb{Z}^d}$ can depend on only a finite number of the coordinates of the points in $(\mathbb{F}^n)^{\mathbb{Z}^d}$. If $x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ and f is a linear functional, then

$$f(x) = \sum_{k=1}^m f_{i_k}(x_{i_k}),$$

where the sum is over a finite set $\{i_k\} \in \mathbb{Z}^n$, each $x_{i_k} \in \mathbb{F}^n$, and each $f_{i_k} \in \widehat{\mathbb{F}^n}$. This means the dual vector space of $(\mathbb{F}^n)^{\mathbb{Z}^d}$ can naturally be identified with the vector space $\mathbb{F}[u^{\pm 1}]^n$. Then $f = \sum_{k=1}^m f_{i_k} u^{i_k} \in \mathbb{F}[u^{\pm 1}]^n$ acts on $x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ as above. If X is a vector space and Y is a subvector space, then there is a vector space homomorphism from the dual vector space \widehat{X} to the dual vector space \widehat{Y} . A linear functional $f \in \widehat{X}$ goes to the linear functional $f' \in \widehat{Y}$ which is the restriction of f to Y . The kernel of this homomorphism is the *annihilator of Y* which is the subspace of \widehat{X} consisting of all linear functional which are identically zero on Y . The annihilator of Y is denoted by Y^\perp . Consequently, the dual vector space \widehat{Y} is isomorphic to \widehat{X}/Y^\perp . This is an observation that we will use repeatedly in what follows. Similarly, if there is a vector space homomorphism φ from the vector space X onto the vector space Y it induces an embedding of \widehat{Y} into \widehat{X} . A linear functional $h \in \widehat{Y}$ goes to the linear functional $h \circ \varphi \in \widehat{X}$. A vector space automorphism φ of the vector space X to itself induces a *dual automorphism* $\widehat{\varphi}$ on the dual space \widehat{X} . As for a homomorphism the linear functional $f \in \widehat{X}$ goes to the linear functional $\widehat{\varphi}(f) = f \circ \varphi \in \widehat{X}$.

For the vector shift $(\mathbb{F}^n)^{\mathbb{Z}^d}$ the coordinate shift σ_j is a vector space automorphism, and its dual automorphism is isomorphic to multiplication by u_j on $\mathbb{F}[u^{\pm 1}]^n$, $\widehat{\sigma}_j(f) \simeq u_j f$. If X is a vector shift contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$, then the dual space \widehat{X} is isomorphic to $\mathbb{F}[u^{\pm 1}]^n$ modulo X^\perp , and the dual automorphism of the shift σ_j restricted to X is the map induced by multiplication by u_j on $\mathbb{F}[u^{\pm 1}]^n/X^\perp$.

Suppose $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ is a vector shift. A *parity check system* for X is defined by a collection of linear functionals f_1, \dots, f_m in $\mathbb{F}[u^{\pm 1}]^n$ with the property that a point $x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ is in X if and only if $f_j(\sigma^m(x)) = 0$ for every $j = 1, \dots, m$ and all $m \in \mathbb{Z}^d$.

For our purposes it will be useful to change our viewpoint slightly. Instead of viewing $(\mathbb{F}^n)^{\mathbb{Z}^d}$ as a vector space with d commuting automorphisms we will think of it as a module over the Laurent polynomial ring $\mathbb{F}[\sigma_1^{\pm 1}, \dots, \sigma_d^{\pm 1}]$. We use the same notational convention as before and denote this ring by $\mathbb{F}[\sigma^{\pm 1}]$. A Laurent polynomial $f = \sum_{k=1}^m \alpha_{i_k} \sigma^{i_k} \in \mathbb{F}[\sigma^{\pm 1}]$ acts on a point $x \in (\mathbb{F}^n)^{\mathbb{Z}^d}$ by $f(x) = \sum_{k=1}^m \alpha_{i_k} \sigma^{i_k}(x)$. Then a vector shift $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ is simply a closed submodule. The dual vector space $\mathbb{F}[u^{\pm 1}]^n$ of $(\mathbb{F}^n)^{\mathbb{Z}^d}$ we now think of as a module over the Laurent polynomial ring $\mathbb{F}[u^{\pm 1}]$ and from now on we refer to it as the *dual module*. Similarly, for a vector shift $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ we refer to the annihilator X^\perp as its *annihilator module* and to the dual space \widehat{X} as the *dual module*. This viewpoint will be very convenient.

3. Definition of a convolutional code. One problem in dimensions greater than one is to formulate a suitable definition of a convolutional code. There are at least three obvious candidates. All three agree in dimension one, and all three are different in higher dimensions. The three candidates are listed below and then discussed with examples to show how they differ.

Condition 1. A d -dimensional convolutional code is a vector shift X contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ whose dual module \widehat{X} contains no nontrivial finite submodules.

Condition 2. A d -dimensional convolutional code is a vector shift X contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ whose dual module \widehat{X} contains no torsion elements.

Condition 3. A d -dimensional convolutional code is a vector shift X contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ whose dual module \widehat{X} is a free module.

Note. In terms of the dynamics of the vector shift, condition 1 is equivalent to the vector shift being transitive; condition 2 is equivalent to the vector shift having strictly positive entropy; and condition 3 is equivalent, as we will see, to the vector shift being algebraically conjugate to a full vector shift.

LEMMA 3.1. *In one-dimension conditions 1, 2, and 3 are equivalent.*

Proof. The Laurent polynomial ring in one variable over a finite field $\mathbb{F}[u^{\pm 1}]$ is a principal ideal domain. The lemma follows immediately from the primary decomposition theorem for finitely generated modules over principal ideal domains. See, for example, [3]. In our case this states that if \mathcal{M} is a finitely generated module over $\mathbb{F}[u^{\pm 1}]$, then it can be written as

$$\bigoplus_{i=1}^m \mathbb{F}[u^{\pm 1}] \oplus \bigoplus_{j=1}^n \mathbb{F}[u^{\pm 1}]/\langle p_j \rangle,$$

where each $p_j \in \mathbb{F}[u^{\pm 1}]$ is a nonzero polynomial. The point for us is that each term $\mathbb{F}[u^{\pm 1}]/\langle p_j \rangle$ is a finite submodule of \mathcal{M} . This follows because any polynomial is equivalent modulo p to a polynomial with degree less than the degree of p . The lemma then follows because if X is a vector shift and its dual module \widehat{X} satisfies condition 1, then there are no $\mathbb{F}[u^{\pm 1}]/\langle p_j \rangle$ terms and it is a free module. \square

The change happens in dimension two because the Laurent polynomial ring in more than one variable over a finite field $\mathbb{F}[\underline{u}^{\pm 1}]$ is no longer a principal ideal domain. However, every ideal is finitely generated, so $\mathbb{F}[\underline{u}^{\pm 1}]$ is a commutative Noetherian ring and modules over these rings are reasonably well behaved.

EXAMPLE 3.1. *First we give an example of a vector shift in two dimensions that satisfies condition 1 but not condition 2. It is a well-known example and is usually called the three dot dynamical system or Ledrappier’s example. Let \mathbb{F}_2 be the finite field with two elements. Let X_L be the vector shift which consists of all $z \in (\mathbb{F}_2)^{\mathbb{Z}^2}$ with*

$$z_{(i,j)} + z_{(i+1,j)} + z_{(i,j+1)} = 0$$

for all $(i, j) \in \mathbb{Z}^2$. The annihilator submodule X_L^\perp of X_L is the submodule of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ generated by the polynomial $1 + u_1 + u_2$. The dual module \widehat{X}_L of X_L is isomorphic to $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]/\langle 1 + u_1 + u_2 \rangle$. This module contains no nontrivial finite submodules because if $p \notin \langle 1 + u_1 + u_2 \rangle$, then two multiples, qp and $q'p$, of p are equivalent if and only if $q + q' \in \langle 1 + u_1 + u_2 \rangle$. Since there are infinitely many Laurent polynomials that pairwise do not add to a multiple of $1 + u_1 + u_2$ any submodule containing p is infinite. On the other hand, every element of the dual module \widehat{X}_L is a torsion element because any element multiplied by the polynomial $1 + u_1 + u_2$ is zero in the module. This means X_L does not satisfy condition 2. A count shows that there are $2n - 1$ admissible $n \times n$ squares in X_L and, consequently, the entropy is zero.

EXAMPLE 3.2. *This example is of a vector shift in two dimensions that satisfies condition 2 but not condition 3. Consider the full two-dimensional shift $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ with the entry in each coordinate of a point z given by $z_{(i,j)} = (z_{(i,j)}^1, z_{(i,j)}^2)$. Let X be the vector shift contained in $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ which consists of all $z \in (\mathbb{F}_2^2)^{\mathbb{Z}^2}$ with*

$$z_{(i,j)}^1 + z_{(i,j+1)}^1 + z_{(i,j)}^2 + z_{(i+1,j)}^2 = 0$$

for all $(i, j) \in \mathbb{Z}^2$. The annihilator module X^\perp in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ is generated by the element

$$\begin{bmatrix} 1 + u_2 \\ 1 + u_1 \end{bmatrix}$$

which means the dual module \widehat{X} is isomorphic to the module

$$\frac{\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2}{\left\langle \begin{bmatrix} 1 + u_2 \\ 1 + u_1 \end{bmatrix} \right\rangle}.$$

We will see that this module is a rank two module with no torsion elements, but it is not a free module. To do this define a map from $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ to the ring $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ by sending the elements

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

to the polynomials $1 + u_1$ and $1 + u_2$, respectively. The point is that the kernel of the map is generated by

$$\begin{bmatrix} 1 + u_2 \\ 1 + u_1 \end{bmatrix},$$

and the image of the map is the ideal $\langle 1 + u_1, 1 + u_2 \rangle$ in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$. So the dual module \widehat{X} is isomorphic to the ideal $\langle 1 + u_1, 1 + u_2 \rangle$ thought of as an $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ module. The polynomials $1 + u_1$ and $1 + u_2$ have no common factors, so the module they generate is rank one if and only if it is all of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$. However, it is clear that 1 is not a linear combination of $1 + u_1$ and $1 + u_2$. Consequently, the dual module \widehat{X} is a rank two module. There are no torsion elements because any nonzero multiple of a nontrivial linear combination of $1 + u_1$ and $1 + u_2$ is not equal to zero. However, it is not free because if p and q are any two generators for $\langle 1 + u_1, 1 + u_2 \rangle$, then $qp + pq = 0$. We have shown that \widehat{X} is a rank two module with no torsion elements, but it is not a free module. This means that the vector shift X satisfies condition 2 but not condition 3. A simple counting argument will show that this vector shift has entropy $\log 2$.

EXAMPLE 3.3. This example is of a vector shift which nontrivially satisfies condition 3. As above consider the full two-dimensional shift $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ with the entry in each coordinate of a point z given by $z_{(i,j)} = (z_{(i,j)}^1, z_{(i,j)}^2)$. Let Y be the vector shift which consists of all $z \in (\mathbb{F}_2^2)^{\mathbb{Z}^2}$ with

$$z_{(i+1,j)}^1 + z_{(i,j+1)}^1 + z_{(i,j)}^2 + z_{(i+1,j)}^2 + z_{(i,j+1)}^2 = 0$$

for all $(i, j) \in \mathbb{Z}^2$. The annihilator module Y^\perp in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ is generated by the element

$$\begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix}$$

which means the dual module \widehat{Y} is isomorphic to the module

$$\frac{\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2}{\left\langle \begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix} \right\rangle}.$$

Using an argument like the one in Example 3.2 we will see that the dual module \widehat{Y} is a free rank one module. Define a map from $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ to $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ by sending the elements

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

to the polynomials $1 + u_1 + u_2$ and $u_1 + u_2$, respectively. The kernel of the map is generated by

$$\begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix},$$

and the image of the map is the ideal $\langle u_1 + u_2, 1 + u_1 + u_2 \rangle$. However, this ideal is clearly all of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$. So the dual module \widehat{Y} is isomorphic to $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ which means Y satisfies condition 3. A simple counting argument will show that this vector shift has entropy $\log 2$.

Condition 1 is clearly not a good choice for the definition of a convolutional code since there are vector shifts satisfying it with zero entropy and it is not possible to encode any information into a zero entropy vector shift. Condition 2 is a possibility, but there are difficulties defining encoders and decoders for vector shifts which satisfy condition 2 but not condition 3. In one dimension algebraic finite state machines are used to define encoders for convolutional codes, but in two or more dimensions there does not appear to be anything except sliding block maps to take their place. For these reasons I have chosen to concentrate on condition 3 and will investigate some of the properties of vector shifts which satisfy this condition.

DEFINITION 3.2. A d -dimensional $[n, k]$ convolutional code is a vector shift X contained in $(\mathbb{F}^n)^{\mathbb{Z}^d}$ with entropy $k \log |\mathbb{F}|$ whose character module \widehat{X} is a free module.

Note. The definition adopted here for a convolutional code is essentially the same as the definition of a free convolutional code in [15] and the definition of a basic convolutional code in [2]. These relationships will be demonstrated by Theorem 4.1.

4. Convolutional codes.

THEOREM 4.1. Let $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ be a d -dimensional vector shift with entropy $\log k$. Then X is an $[n, k]$ convolutional code if and only if any one, and hence all, of the following conditions hold.

1. The dual module \widehat{X} of X has rank k , and $\mathbb{F}[\underline{u}^{\pm 1}]^n$ is isomorphic to the direct sum of the annihilator module X^\perp of X and the dual module \widehat{X} of X .
2. The dual module \widehat{X} has rank k , and there exists a parity check system for X defined by $n - k$ linear functionals on $(\mathbb{F}^n)^{\mathbb{Z}^d}$.
3. The vector shift X is algebraically conjugate to $(\mathbb{F}^k)^{\mathbb{Z}^d}$.
4. There are n linear functionals $h^1, \dots, h^n \in \mathbb{F}[\underline{u}^{\pm 1}]^k$ with $\langle h^1, \dots, h^n \rangle = \mathbb{F}[\underline{u}^{\pm 1}]^k$ and the map defined by (h^1, \dots, h^n) from $(\mathbb{F}^k)^{\mathbb{Z}^d}$ into $(\mathbb{F}^n)^{\mathbb{Z}^d}$ has image X . The map (h^1, \dots, h^n) defines a convolutional encoder for X , and we construct its decoder.
5. There are k linear functionals $f^1, \dots, f^k \in \mathbb{F}[\underline{u}^{\pm 1}]^n$ with $(\cap \ker f^i) \cap X = \{0\}$. The map defined by (f^1, \dots, f^k) from $(\mathbb{F}^n)^{\mathbb{Z}^d}$ into $(\mathbb{F}^k)^{\mathbb{Z}^d}$ defines a convolutional decoder for X , and we construct its encoder.

Note. In one dimension a convolutional encoder is usually defined by an algebraic finite state automaton. An invertible algebraic map from $(\mathbb{F}^k)^{\mathbb{Z}}$ into $(\mathbb{F}^n)^{\mathbb{Z}}$ is a special case of such a convolutional encoder. In dimensions higher than one it is not clear

how to use something similar to a finite state automaton to define a convolutional encoder. In this work a *multidimensional convolutional encoder* is considered to be an invertible algebraic map from $(\mathbb{F}^k)^{\mathbb{Z}^d}$ into $(\mathbb{F}^n)^{\mathbb{Z}^d}$. Similarly, a *multidimensional convolutional decoder* is considered to be an algebraic map from $(\mathbb{F}^n)^{\mathbb{Z}^d}$ onto $(\mathbb{F}^k)^{\mathbb{Z}^d}$ which restricted to the convolutional code $X \subseteq (\mathbb{F}^n)^{\mathbb{Z}^d}$ is invertible.

Before proving the theorem we state a theorem and prove a lemma. First we state a version of the Quillen–Suslin theorem which is the solution to the Serre conjecture [9].

THEOREM 4.2. *Suppose M and N are submodules of $\mathbb{F}[\underline{u}^{\pm 1}]^n$ and $\mathbb{F}[\underline{u}^{\pm 1}]^n = M \oplus N$; then M and N are free modules.*

LEMMA 4.3. *Let $M \subseteq \mathbb{F}[\underline{u}^{\pm 1}]^n$ be a submodule. Then $\mathbb{F}[\underline{u}^{\pm 1}]^n/M$ is a free module if and only if $\text{rank } M + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/M = n$.*

Proof. First suppose $\mathbb{F}[\underline{u}^{\pm 1}]^n/M$ is a free module. Then the Quillen–Suslin theorem, Theorem 4.2 shows that M is a free module and that $\mathbb{F}[\underline{u}^{\pm 1}]^n$ is isomorphic to the direct sum of M and $\mathbb{F}[\underline{u}^{\pm 1}]^n/M$, so $\text{rank } M + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/M = n$.

Suppose $\text{rank } M + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/M = n$. Let $\{x^1, \dots, x^\ell\}$ be a minimal generating set for M and $\{x^{\ell+1}, \dots, x^n\} \subseteq \mathbb{F}[\underline{u}^{\pm 1}]^n$ be a set whose M -equivalence classes generate $\mathbb{F}[\underline{u}^{\pm 1}]^n/M$. Then $\{x^1, \dots, x^n\}$ generates $\mathbb{F}[\underline{u}^{\pm 1}]^n$. A generating set for $\mathbb{F}[\underline{u}^{\pm 1}]^n$ with n elements is a base. If $\sum_{j=\ell+1}^n \alpha_j x^j \in M$, then the sum must be zero and so every α_j is zero. This means the equivalence classes of $x^{\ell+1}, \dots, x^n$ are linearly independent in $\mathbb{F}[\underline{u}^{\pm 1}]^n/M$ and so form a base. \square

Proof of Theorem 4.1. Proof of 1. Statement 1 is equivalent to the definition of a convolutional code by the Quillen–Suslin theorem, Theorem 4.2.

Proof of 2. Suppose \widehat{X} has rank k and there exists a parity check system for X defined by $n - k$ linearly independent linear functionals on $\mathbb{F}[\underline{u}^{\pm 1}]^n$. Then $\text{rank } X^\perp \leq n - k$, but we know $\text{rank } \widehat{X} + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp \geq n$. This forces $\text{rank } X^\perp = n - k$ and $\text{rank } \widehat{X} + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp = n$. By Lemma 4.3 $\mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp$ is a free module.

Conversely, suppose $\mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp$ is a free module. Then by Lemma 4.3 $\text{rank } X^\perp + \text{rank } \mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp = n$. So X^\perp is a free module with rank $n - k$, and a base of $n - k$ linear functional for X^\perp defines a parity check system for X .

Proof of 3. This follows immediately because on the one hand an algebraic conjugacy between X and $(\mathbb{F}^k)^{\mathbb{Z}^d}$ induces an isomorphism between their dual modules. Since the dual module of $(\mathbb{F}^k)^{\mathbb{Z}^d}$ is $\mathbb{F}[\underline{u}^{\pm 1}]^k$ the dual module \widehat{X} is free. On the other hand, if the dual module \widehat{X} of X is free and X has entropy $\log k$, then \widehat{X} is isomorphic to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. An isomorphism between these two dual modules induces an algebraic conjugacy between the vector shifts X and $(\mathbb{F}^k)^{\mathbb{Z}^d}$. So X is a d -dimensional convolutional code.

A more constructive proof of the second statement is to first observe that the dual module \widehat{X} is a free rank k module. Take any base $\{f^1, \dots, f^k\}$ for it and use this base to define an algebraic conjugacy f from X to $(\mathbb{F}^k)^{\mathbb{Z}^d}$ by $f(x)_{\underline{m}} = (f^1(\underline{\sigma}^{\underline{m}}(x)), \dots, f^k(\underline{\sigma}^{\underline{m}}(x)))$ for each $\underline{m} \in \mathbb{Z}^d$. This map is clearly a continuous, shift-commuting, algebraic map from X into $(\mathbb{F}^k)^{\mathbb{Z}^d}$, it is one-to-one because $\{f^1, \dots, f^k\}$ is a base for \widehat{X} , and it is onto because the two spaces have the same entropy.

Proof of 4. Suppose the map $h = (h^1, \dots, h^n)$ defined as in the proof of statement 1 has image X . The map is continuous, shift-commuting, and algebraic. We need only to see that it is one-to-one, and we will have proved that X is algebraically conjugate to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. To do this we define an inverse. Let $e^i \in \mathbb{F}[\underline{u}^{\pm 1}]^k$ be the i th

coordinate linear functional; that is, $e^i(x) = x_i$. Then the map defined from $(\mathbb{F}^k)^{\mathbb{Z}^d}$ to itself by $e = (e^1, \dots, e^k)$ is the identity. Since $\langle h^1, \dots, h^n \rangle = \mathbb{F}[\underline{u}^{\pm 1}]^k$ there exist for each $i = 1, \dots, k$, elements $p_1^i, \dots, p_n^i \in \mathbb{F}[\underline{u}^{\pm 1}]$ with $p_1^i h^1 + \dots + p_n^i h^n = e^i$. Let $p^i = (p_1^i, \dots, p_n^i) \in \mathbb{F}[\underline{u}^{\pm 1}]^n$. Then $p^i \circ h = e^i \in \mathbb{F}[\underline{u}^{\pm 1}]^k$. Define the map $p = (p^1, \dots, p^k)$ from $\mathbb{F}[\underline{u}^{\pm 1}]^n$ to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. We compute to see that the map p is the inverse of the map h :

$$\begin{aligned} (p \circ h(x))_{\underline{m}} &= p(h(\underline{\sigma}^{\underline{m}}(x))) \\ &= (p^1(h(\underline{\sigma}^{\underline{m}}(x)), \dots, p^k(h(\underline{\sigma}^{\underline{m}}(x)))) \\ &= (e^1(\underline{\sigma}^{\underline{m}}(x)), \dots, e^k(\underline{\sigma}^{\underline{m}}(x))) \\ &= e(\underline{\sigma}^{\underline{m}}(x)) \\ &= x_{\underline{m}}. \end{aligned}$$

This shows that X is algebraically conjugate to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. The map h is a convolutional encoder for X , and the map p is its decoder.

Conversely, if X is an $[n, k]$ convolutional code, then it is a vector shift in $\mathbb{F}[\underline{u}^{\pm 1}]^n$ which is algebraically conjugate to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. Let h be an algebraic conjugacy from $\mathbb{F}[\underline{u}^{\pm 1}]^k$ to $X \subseteq \mathbb{F}[\underline{u}^{\pm 1}]^n$. Then it is defined by n coordinate functions, $h = (h^1, \dots, h^n)$, and each h^j is a linear functional from $\mathbb{F}[\underline{u}^{\pm 1}]^k$. Since h is invertible $\langle h^1, \dots, h^n \rangle = \mathbb{F}[\underline{u}^{\pm 1}]^k$.

Proof of 5. Suppose there are k linear functionals $f^1, \dots, f^k \in \mathbb{F}[\underline{u}^{\pm 1}]^n$ with $(\cap \ker f^i) \cap X = \{0\}$. Then $f = (f^1, \dots, f^k)$ defines a map from $\mathbb{F}[\underline{u}^{\pm 1}]^n$ into $\mathbb{F}[\underline{u}^{\pm 1}]^k$ as before. The map restricted to X is one-to-one since it has kernel $\{0\}$. It takes X onto $\mathbb{F}[\underline{u}^{\pm 1}]^k$ by entropy considerations and so defines an algebraic conjugacy between X and $\mathbb{F}[\underline{u}^{\pm 1}]^k$. This means X is a convolutional code. Next we define the encoder for this decoder. Since f defines an algebraic conjugacy, the X^\perp -equivalence classes of f^1, \dots, f^k generate $\mathbb{F}[\underline{u}^{\pm 1}]^n/X^\perp$. This means that for each $j = 1, \dots, n$ there are elements $q_1^j, \dots, q_k^j \in \mathbb{F}[\underline{u}^{\pm 1}]$ with $q_1^j f^1 + \dots + q_k^j f^k = e^j$ when restricted to X , where e^j is the j th coordinate functional as defined in the proof of statement 4. As in the proof of statement 4 let $q^j = (q_1^j, \dots, q_k^j) \in \mathbb{F}[\underline{u}^{\pm 1}]^k$. Define the map $q = (q^1, \dots, q^n)$ from $\mathbb{F}[\underline{u}^{\pm 1}]^k$ to $\mathbb{F}[\underline{u}^{\pm 1}]^n$. Compute as in the proof of statement 4 to see that the composition $q \circ f$ is the identity when restricted to X . The map q is a convolutional encoder for X , and the map f is its decoder.

Conversely, if X is an $[n, k]$ convolutional code, then it is a vector shift in $\mathbb{F}[\underline{u}^{\pm 1}]^n$ which is algebraically conjugate to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. Let f be an algebraic conjugacy from $X \subseteq \mathbb{F}[\underline{u}^{\pm 1}]^n$ to $\mathbb{F}[\underline{u}^{\pm 1}]^k$. Then it is defined by k coordinate functions, $f = (f^1, \dots, f^k)$, and each f^j is a linear functional from $\mathbb{F}[\underline{u}^{\pm 1}]^n$. Since f is invertible $(\cap \ker f^i) \cap X = \{0\}$. \square

5. Examples. Here we will examine two two-dimensional convolutional codes. We will see how each statement of Theorem 4.1 applies.

EXAMPLE 5.1. *This is the two-dimensional $[2, 1]$ convolutional code of Example 3.3. As we noted in the example the annihilator module Y^\perp in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ is the rank one free module generated by the element*

$$\begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix}$$

and the dual module \widehat{Y} is isomorphic to the free rank one module

$$\frac{\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2}{\left\langle \begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix} \right\rangle}.$$

To see that $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ splits as stated in the first assertion of Example 3.3 let W be the free rank one submodule of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ generated by the element

$$\begin{bmatrix} 1 + u_1 + u_2 \\ u_1 + u_2 \end{bmatrix}.$$

We will see that this linear functional comes from the encoding map. Now observe that $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ can be written as $Y^\perp \oplus W$ since $Y^\perp \cap W = \{0\}$ and

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix} + \begin{bmatrix} 1 + u_1 + u_2 \\ u_1 + u_2 \end{bmatrix},$$

while

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix} + (u_1 + u_2) \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

To see assertion 2 we simply let the generator for the annihilator module

$$\begin{bmatrix} u_1 + u_2 \\ 1 + u_1 + u_2 \end{bmatrix}$$

define the parity check system from $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ to $(\mathbb{F}_2)^{\mathbb{Z}^2}$.

Define an algebraic conjugacy h from $(\mathbb{F}_2)^{\mathbb{Z}^2}$ to Y by

$$h(x)_{(i,j)} = (x_{(i,j)} + x_{(i+1,j)} + x_{(i,j+1)}, x_{(i+1,j)} + x_{(i,j+1)})$$

for all $(i, j) \in \mathbb{Z}^2$. The map is onto and one-to-one since the inverse on Y is given by

$$h^{-1}(z)_{(i,j)} = z_{(i,j)}^1 + z_{(i,j)}^2$$

for all $(i, j) \in \mathbb{Z}^2$, where the (i, j) th coordinate of the point z is $z_{(i,j)} = (z_{(i,j)}^1, z_{(i,j)}^2)$.

This shows directly that Y is algebraically conjugate to $(\mathbb{F}_2)^{\mathbb{Z}^2}$ and satisfies assertion 3.

The two linear functionals in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$ which satisfy assertion 4 are the ones used to define the map h ; that is,

$$h^1 = 1 + u_1 + u_2 \quad \text{and} \quad h^2 = u_1 + u_2.$$

This map produces the subspace used in statement 1.

The linear functional in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$ which satisfies assertion 5 is the one used to define the map h^{-1} , namely,

$$f^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

EXAMPLE 5.2. Next is a two-dimensional $[3, 2]$ convolutional code over the finite field \mathbb{F}_2 . To define the convolutional code X we use statement 4 of Theorem 4.1. Let

$$h^1 = \begin{bmatrix} 1 + u_1 \\ 1 + u_2 \end{bmatrix}, \quad h^2 = \begin{bmatrix} u_1 + u_1u_2 + u_1^2u_2 \\ 1 + u_2 + u_1u_2 + u_1u_2^2 \end{bmatrix},$$

and

$$h^3 = \begin{bmatrix} u_1 + u_2 + u_1^2 + u_1u_2 \\ 1 + u_1 + u_2 + u_2^2 + u_1u_2 \end{bmatrix}$$

be elements of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$. Then note that

$$(1 + u_1u_2)h^1 + h^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad (u_1 + u_2)h^1 + h^3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

So the three linear functionals generate all of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$. It can also be seen that no two of them generate $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$. We define the encoder to be the map $h = (h^1, h^2, h^3)$ from $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ into $(\mathbb{F}_2^3)^{\mathbb{Z}^2}$. Then let X be the image of the map h . Next we construct the decoder for this encoder using the previous equations for

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Set

$$f^1 = \begin{bmatrix} 1 + u_1u_2 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad f^2 = \begin{bmatrix} u_1 + u_2 \\ 0 \\ 1 \end{bmatrix}$$

which are two linear functionals in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3$. They define a map $f = (f^1, f^2)$ from $(\mathbb{F}_2^3)^{\mathbb{Z}^2}$ to $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ which is the inverse of the map h when restricted to X . The linear functionals f_1 and f_2 satisfy statement 5 of the theorem.

The two maps h and f show that X is algebraically conjugate to $(\mathbb{F}_2^2)^{\mathbb{Z}^2}$ satisfying statement 3.

Next we exhibit the parity check system. It is defined by a single linear functional in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3$. To find it we solve the equation

$$\begin{bmatrix} 1 + u_1 & u_1 + u_1u_2 + u_1^2u_2 & u_1 + u_2 + u_1^2 + u_1u_2 \\ 1 + u_2 & 1 + u_2 + u_1u_2 + u_1u_2^2 & 1 + u_1 + u_2 + u_2^2 + u_1u_2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where α , β , and γ are elements of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]$. The matrix on the left describes the map h . The result is the linear functional

$$\begin{bmatrix} u_2(1 + u_2 + u_1^2) \\ 1 + u_1 \\ 1 + u_2 \end{bmatrix}$$

in $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3$, and it defines the parity check system.

The final step is to examine assertion 1. We know that X^\perp is the subspace of $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3$ spanned by

$$\begin{bmatrix} u_2(1 + u_2 + u_1^2) \\ 1 + u_1 \\ 1 + u_2 \end{bmatrix}.$$

Let W be the subspace

$$\left\langle \left[\begin{array}{c} 1 + u_1 \\ u_1 + u_1 u_2 + u_1^2 u_2 \\ u_1 + u_2 + u_1^2 + u_1 u_2 \end{array} \right], \left[\begin{array}{c} 1 + u_2 \\ 1 + u_2 + u_1 u_2 + u_2^2 \\ 1 + u_1 + u_2 + u_2^2 + u_1 u_2 \end{array} \right] \right\rangle.$$

The subspace W is determined by the map h . We see that W is isomorphic to $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3 / X^\perp$ which is in turn isomorphic to $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^2$. And finally we observe that $\mathbb{F}_2[u_1^{\pm 1}, u_2^{\pm 1}]^3$ can be written as $X^\perp \oplus W$.

Acknowledgment. I would like to thank Barry Trager for his help with the algebra used in this work.

REFERENCES

- [1] R. ADLER, D. COPPERSMITH, AND M. HASSNER, *Algorithms for sliding block codes*, IEEE Trans. Inform. Theory, 29 (1983), pp. 5–22.
- [2] E. FORNASISI AND M. E. VALCHER, *Algebraic aspects of two-dimensional convolutional codes*, IEEE Trans. Inform. Theory, 40 (1994), pp. 1068–1082.
- [3] N. JACOBSON, *Basic Algebra I*, W. H. Freeman, New York, 1985.
- [4] B. KITCHENS, *Symbolic Dynamics, One-sided, Two-sided and Countable State Markov Shifts*, Springer-Verlag, Berlin, 1998.
- [5] B. KITCHENS, *Symbolic dynamics and convolutional codes*, in Codes, Systems, and Graphical Models, IMA Vol. Math. Appl. 123, Springer-Verlag, New York, 2001, pp. 347–360.
- [6] T. LAM, *Serre's Conjecture*, Lecture Notes in Math. 635, Springer-Verlag, Berlin, 1978.
- [7] S. LANG, *Algebra*, Springer-Verlag, Heidelberg, 2002.
- [8] D. LIND AND B. MARCUS, *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, Cambridge, UK, 1995.
- [9] A. LOGAR AND B. STURMFELS, *Algorithms for the Quillen-Suslin theorem*, J. Algebra, 145 (1992), pp. 231–239.
- [10] B. MARCUS AND J. ROSENTHAL, EDs., *Codes, Systems, and Graphical Models*, IMA Vol. Math. Appl. 123, Springer-Verlag, New York, 2001.
- [11] B. H. MARCUS, P. H. SIEGEL, AND J. K. WOLF, *Finite-state modulation codes for data storage*, IEEE J. Sel. Areas Comm., 10 (1992), pp. 5–37.
- [12] R. McELIECE, *The algebraic theory of convolutional codes*, in Handbook of Coding Theory, V. Pless and W. Hoffman, eds., North-Holland, Amsterdam, 1998, pp. 1065–1138.
- [13] K. SCHMIDT, *Dynamical Systems of Algebraic Origin*, Birkhäuser, Basel, 1995.
- [14] K. SCHMIDT, *Multi-dimensional dynamical systems*, in Codes, Systems, and Graphical Models, IMA Vol. Math. Appl. 123, Springer-Verlag, New York, 2001, pp. 67–82.
- [15] P. WEINER, *Basic properties of multidimensional convolutional codes*, in Codes, Systems, and Graphical Models, IMA Vol. Math. Appl. 123, Springer-Verlag, New York, 2001, pp. 397–414.

INTERFERENCE PATTERNS IN REGULAR GRAPHS WITH BIJECTIVE COLORINGS*

PETER C. FISHBURN[†] AND PAUL E. WRIGHT[†]

Abstract. Let $\mathcal{G}_d(n)$ be the set of d -regular simple graphs with n vertices, and for $G = (V, E)$ in $\mathcal{G}_d(n)$ let $f : V \rightarrow \{1, 2, \dots, n\}$ be a bijective coloring of the vertices of G . Also let α denote an interference parameter in $\{0, 1, \dots, n - 1\}$, and define the interference number of f with respect to G and α as the number of edges $\{u, v\}$ in E for which $\min\{|f(u) - f(v)|, n - |f(u) - f(v)|\} \leq \alpha$. We consider two interference number problems for feasible (n, α, d) . The first is to specify a $G \in \mathcal{G}_d(n)$ and an f for which the interference number is as small as possible. The second is to determine a $G \in \mathcal{G}_d(n)$ whose minimum interference number over all f is as large as possible. A previous paper completely solves both problems for $d = 2$. The present paper solves the first problem for all $d \geq 3$ and obtains partial results for the second problem for $d \geq 3$ that focus on interference-minimizing f 's when G consists of disjoint copies of K_{d+1} .

Key words. bijective graph coloring, regular graphs, interference, channel assignment

AMS subject classifications. 05B99, 05C35

PII. S0895480100382342

1. Introduction. This paper is the third in a series concerned with minimizing interference in vertex colorings of regular graphs. It and its predecessors [1, 2] are motivated by telecommunications problems such as the design of planar regions for cellular telephone networks and the assignment of frequencies to regions. Our graph abstraction represents regions by vertices, contiguous regions by edges, and frequencies or frequency classes by colors. We assume that the number of colors available equals the number n of regions or vertices, and that colors i and j in the color set $\{1, 2, \dots, n\}$ interfere if and only if the distance between them, reckoned circularly [1, 4, 9] as

$$D(i, j) = \min\{|i - j|, n - |i - j|\},$$

is less than or equal to an interference parameter α . Given a coloring f from the vertex set V into $\{1, 2, \dots, n\}$, interference occurs between regions u and v if they are neighbors, or $\{u, v\}$ is in the edge set E , and $D(f(u), f(v)) \leq \alpha$.

The focus of [1] was on minimization of interference without regard to capacity and therefore placed no restrictions on colorings. In [2] and the present paper, colorings are assumed to be bijections between V and $\{1, 2, \dots, n\}$ so that they maximize capacity when capacity is assessed by the number of different colors assigned to vertices. Additional background on our approach as well as related coloring problems motivated by the channel assignment problem of Hale [5] appears in [1, 2] and their references, including [3, 6, 7, 8].

As in [2], let $\mathcal{G}_d(n)$ for $n > d \geq 2$ be the set of d -regular simple graphs with n vertices. It is assumed that dn is even so that $\mathcal{G}_d(n)$ is not empty. We refer to a bijective coloring as an *assignment* and let S_n denote the set of maps f from V onto $\{1, 2, \dots, n\}$ when $|V| = n$. The *interference number* $\mu_\alpha(G, f)$ of $G = (V, E)$ in $\mathcal{G}_d(n)$ for assignment f and interference threshold $\alpha \in \{0, 1, \dots, n - 1\}$ is the number of

*Received by the editors December 11, 2000; accepted for publication (in revised form) March 19, 2002; published electronically May 23, 2002.

<http://www.siam.org/journals/sidma/15-3/38234.html>

[†]AT&T Labs-Research, Florham Park, NJ 07932 (fish@research.att.com).

edges in E whose vertex colors interfere:

$$\mu_\alpha(G, f) = |\{\{u, v\} \in E : D(f(u), f(v)) \leq \alpha\}|.$$

We focus on two interference number minimization problems for assignments that, given (n, α, d) , are concerned with the quantities

$$m(n, \alpha, d) = \min_{G \in \mathcal{G}_d(n)} \min_{f \in \mathcal{S}_n} \mu_\alpha(G, f),$$

$$M(n, \alpha, d) = \max_{G \in \mathcal{G}_d(n)} \min_{f \in \mathcal{S}_n} \mu_\alpha(G, f).$$

Problem I. Determine $m(n, \alpha, d)$ along with graphs in $\mathcal{G}_d(n)$ and their assignments f for which $\mu_\alpha(G, f) = m(n, \alpha, d)$.

Problem II. Determine $M(n, \alpha, d)$ along with graphs in $\mathcal{G}_d(n)$ and their assignments f for which $\mu_\alpha(G, f) = M(n, \alpha, d)$.

In Problem I, a system designer gets to choose both the graph and its assignment to minimize interference. In Problem II, the designer selects an interference-minimizing assignment after an adversary chooses a graph that will maximize the minimum interference number. The quantities m and M bracket the minimum interference numbers of all graphs in $\mathcal{G}_d(n)$ for threshold α . By the definitions,

$$m(n, \alpha, d) \leq \min_{f \in \mathcal{S}_n} \mu_\alpha(G, f) \leq M(n, \alpha, d)$$

for all $G \in \mathcal{G}_d(n)$.

Solutions for both problems are given in [2] when $d = 2$. Obviously, every $G \in \mathcal{G}_2(n)$ is the union of disjoint cycles. We noted in [2] that $m(n, \alpha, 2)$ is always realized by the cycle C_n on all n vertices with

$$m(n, \alpha, 2) = \begin{cases} 0 & \text{if } \alpha < \frac{n}{2} - 1, \\ \frac{n}{2} & \text{if } \alpha = \frac{n}{2} - 1 \quad (\text{so } n \text{ is even}), \\ n & \text{if } \alpha \geq \lfloor \frac{n}{2} \rfloor. \end{cases}$$

Our first new result extends this to higher-degree regular graphs.

THEOREM 1.1. *Suppose $d \geq 3$. Then $m(n, \alpha, d)$ is always realized by a connected graph in $\mathcal{G}_d(n)$ with*

$$m(n, \alpha, d) = \begin{cases} 0 & \text{if } \alpha \leq \frac{n-d-1}{2}, \\ \frac{nd}{2} & \text{if } \alpha \geq \lfloor \frac{n}{2} \rfloor. \end{cases}$$

In addition,

$$m(n, \alpha, d) = \frac{n(j+1)}{2} \quad \text{if } \alpha = \frac{n-d+j}{2}$$

for

$$j \in \{1, 3, 5, \dots, d-2\} \text{ when } d \text{ is odd and } n \text{ is even,}$$

$$j \in \{1, 3, 5, \dots, d-3\} \text{ when } d \text{ is even and } n \text{ is odd,}$$

$$j \in \{0, 2, 4, \dots, d-2\} \text{ when } d \text{ and } n \text{ are both even.}$$

Theorem 1.1 is proved in the next section. Graphs in $\mathcal{G}_d(n)$ and assignments that realize the m values are described there.

Theorems 2.5 and 2.6 in [2] identify $M(n, \alpha, 2)$ values and their realizing graphs and assignments for $d = 2$. In sharp contrast to connectedness for m , $M(n, \alpha, 2)$ can always be realized by a graph in $\mathcal{G}_d(n)$ that has as many components as possible. The adversary can never do better than to include as many disjoint copies of C_3 as possible in G , as seen in the following examples:

- (i) If $n \equiv 0 \pmod{3}$ and $\lfloor n/3 \rfloor \leq \alpha < \lfloor n/2 \rfloor - 1$, then $M(n, \alpha, 2) = n/3$, and this is realized only by the graph composed of $n/3$ disjoint copies of C_3 .
- (ii) If $n \equiv 2 \pmod{3}$, then $M(n, \alpha, 2) = (n - 5)/3$ if $\lfloor n/3 \rfloor \leq \alpha < \lfloor 2n/5 \rfloor$, and $M(n, \alpha, 2) = (n - 2)/3$ if $\lfloor 2n/5 \rfloor \leq \alpha < \lfloor n/2 \rfloor - 1$. These values of M are realized only by a G that has $(n - 5)/3$ disjoint copies of C_3 and one copy of C_5 .

The proof of Theorems 2.5 and 2.6 in [2] requires detailed case analyses. Rather than attempt such analyses here for $d \geq 3$, we will be guided by the conjecture that $M(n, \alpha, d)$ for $d \geq 3$ can always be realized by a graph in $\mathcal{G}_d(n)$ that includes as many copies as possible of the complete graph K_{d+1} on $d + 1$ vertices. The conjecture is supported not only by the results for $d = 2$ but also by the fact that complete subgraphs on $d + 1$ vertices often force interference that can be avoided in larger d -regular components.

Our most complete M -type result for $d > 2$ occurs for $d = 3$, where n must be even and every 3-regular component of G must have an even number of vertices. We restrict our analysis to graphs in $\mathcal{G}_3(n)$ with as many components as possible, i.e.,

- (a) if $n \equiv 0 \pmod{4}$, then G consists of $n/4$ disjoint copies of K_4 ;
- (b) if $n \equiv 2 \pmod{4}$, then G consists of $(n - 6)/4$ disjoint copies of K_4 and one 6-vertex component.

There are two graphs in $\mathcal{G}_3(6)$, and hence two options for the 6-vertex component in (b). We refer to these as H_1 and H_2 ; see Figure 1.1. The adversary's choice between H_1 and H_2 is immaterial in many cases but not in others. When $(n, \alpha) = (6, 1)$, H_1 forces at least two edges whose colors interfere, whereas H_2 has an assignment with no interference. We illustrate this in the second row of Figure 1.1. We refer to edges with interference as *bad edges* and draw them thicker than the others. When $(n, \alpha) = (14, 4)$, every C_3 has a bad edge, so H_2 in the bottom row of Figure 1.1 forces six bad edges, whereas H_1 forces only four.

For the following theorem, $M^*(n, \alpha, 3)$ is defined like $M(n, \alpha, 3)$ when $\mathcal{G}_3(n)$ is replaced by the special subset noted above for (a) or (b).

THEOREM 1.2. *Suppose $d = 3$ and $k \geq 1$. If $n = 4k$, then $M^*(n, \alpha, 3)$ equals*

$$\begin{aligned}
 0 & \text{ if } \alpha \leq k - 1, \\
 k & \text{ if } k \leq \alpha \leq \lceil 4k/3 \rceil - 2, \\
 2k & \text{ if } \lceil 4k/3 \rceil - 1 \leq \alpha \leq 2k - 2, \\
 4k & \text{ if } \alpha = 2k - 1, \\
 6k & \text{ if } \alpha \geq 2k.
 \end{aligned}$$

If $n = 4k + 2$, then $M^*(n, \alpha, 3)$ equals

- 0 if $\alpha \leq k - 1$,
- 2 if $(k, \alpha) \in \{(1, 1), (2, 2)\}$,
- $k - 1$ if $k \geq 3, k \leq \alpha \leq \lceil 4k/3 \rceil - 1$, and it is false that both $k - 1 \equiv 0 \pmod{3}$ and $\alpha = \lceil 4k/3 \rceil - 1$,
- $2(k - 1)$ if $k \geq 4, k - 1 \equiv 0 \pmod{3}$ and $\alpha = \lceil 4k/3 \rceil - 1$,
- $2k$ if $\lceil 4k/3 \rceil \leq \alpha \leq 2k - 1$,
- $4k + 2$ if $\alpha = 2k$,
- $6k + 3$ if $\alpha \geq 2k + 1$.

We defer the proof of Theorem 1.2 to section 4 because its first part uses a result for $d \geq 3$ that is proved in section 3. That result assumes that n is a multiple of $d + 1$ and that G consists of disjoint copies of K_{d+1} . Let

kK_{d+1} denote the graph that consists of k copies of K_{d+1} ,

and let

$$N(k, \alpha, d) = \min_{f \in S_{k(d+1)}} \mu_\alpha(kK_{d+1}, f).$$

THEOREM 1.3. *Suppose $d \geq 3$ and $k \in \{1, 2, \dots\}$ with $n = k(d + 1)$. Then $N(k, \alpha, d) = 0$ if and only if $\alpha \leq k - 1$, and $N(k, \alpha, d) = kd(d + 1)/2$ if and only if $\alpha \geq \lfloor n/2 \rfloor = \lfloor k(d + 1)/2 \rfloor$. Moreover, if $t \in \{1, 2, \dots\}$, $d + 1 \geq 2t$, and*

$$\left\lfloor \frac{(d + 1)(k - 1)}{d + 2 - t} \right\rfloor + 1 \leq \alpha \leq \left\lfloor \frac{(d + 1)(k - 1)}{d + 1 - t} \right\rfloor,$$

then $N(k, \alpha, d) = tk$.

This is proved in section 3 where we give an explicit assignment for kK_{d+1} for the part of the theorem that involves t . With respect to that part, the bounds of Theorem 1.3 on α along with $d + 1 \geq 2t$ imply that the minimum number of bad edges in a K_{d+1} with the fewest bad edges equals t . This occurs, for example, when the vertices of a particular K_{d+1} are colored $1, 2, \alpha + 3, \alpha + 4, 2\alpha + 5, 2\alpha + 6, \dots$, with t instances of $|f(u) - f(v)| = 1$. The theorem asserts that when this is true, there is an assignment from the $k(d + 1)$ vertices of kK_{d+1} onto $\{1, 2, \dots, k(d + 1)\}$ such that every copy of K_{d+1} has only t bad edges.

The $d + 1 \geq 2t$ constraint in the latter part of Theorem 1.3 limits α coverage. With $t^* = \lfloor (d + 1)/2 \rfloor$, the α range in the theorem goes from 0 to $\lfloor (d + 1)(k - 1)/(d + 1 - t^*) \rfloor$, or approximately $2(k - 1)$, and then jumps to $\alpha \geq \lfloor k(d + 1)/2 \rfloor$ where all edges are bad. When $d = 3$, this misses only $\alpha = 2k - 1$, which is covered in the first part of Theorem 1.2 by the line for $\alpha = 2k - 1$. The gap of omitted α 's is larger for $d \geq 4$, but Theorem 1.3's coverage up to about $2(k - 1)$ may well include the most likely interference thresholds encountered in practice. Nevertheless, it would be nice to have definitive results for the missing α range, and we encourage further work along this line.

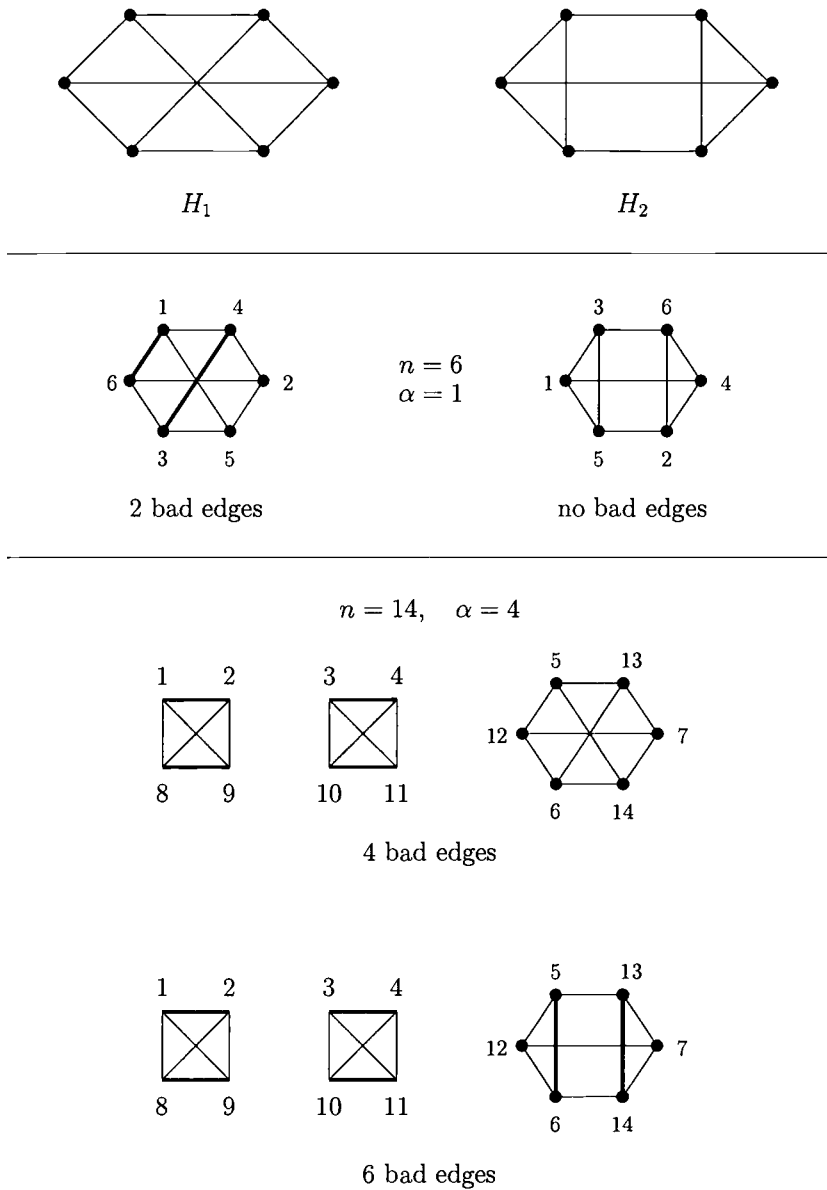


FIG. 1.1. $d = 3, n \equiv 2 \pmod{4}$.

We conclude this introduction with a version of the latter part of Theorem 1.3 that may be interesting in its own right as a purely graph-theoretic result, and with a conjecture that could motivate further research on bijective colorings of regular graphs.

PROPOSITION 1.4. *Suppose k, d, t , and α are positive integers that satisfy $k \geq 1$,*

$d \geq 2, 1 \leq t \leq (d + 1)/2$, and

$$\left\lfloor \frac{(d + 1)(k - 1)}{d + 2 - t} \right\rfloor + 1 \leq \left\lfloor \frac{(d + 1)(k - 1)}{d + 1 - t} \right\rfloor = \alpha.$$

If a simple graph G consists of k disjoint copies of K_{d+1} , then there is a bijection f from G 's vertices onto $\{1, 2, \dots, k(d + 1)\}$ such that every copy of K_{d+1} in G has no more than t edges $\{u, v\}$ for which $D(f(u), f(v)) \leq \alpha$.

CONJECTURE 1.5. Given the hypotheses of Proposition 1.4, if G is a simple d -regular graph with $k(d + 1)$ vertices, then there is a bijection f from G 's vertices onto $\{1, 2, \dots, k(d + 1)\}$ such that no more than tk edges $\{u, v\}$ of G have $D(f(u), f(v)) \leq \alpha$.

2. Proof of Theorem 1.1. Assume that $d \geq 3$. With $|V| = n > d$ and nd even, we color the vertices $1, 2, \dots, n$ and define edges for $G \in \mathcal{G}_d(n)$ to minimize the number of bad edges. It will be clear that G is connected.

Suppose that either d is odd and n is even or d is even and n is odd. Let $V_d(i)$ be the set of d vertices j for which $D(i, j)$, i.e., $\min\{|i - j|, n - |i - j|\}$, is as large as possible. For example, if $d = 3, n = 10$, and $i = 1$, then $V_3(1) = \{5, 6, 7\}$; if $d = 4, n = 13$, and $i = 1$, then $V_4(1) = \{6, 7, 8, 9\}$. Take $\{i, j\} \in E$ if $j \in V_d(i)$ or, equivalently, if $i \in V_d(j)$ for all i and j . Regardless of $\alpha, G = (V, E)$ with the noted assignment minimizes the number of bad edges. We count the number of bad edges for the two cases of opposite parities for d and n .

Suppose d is odd and n is even. Then

$$V_d(1) = \left\{ \frac{n}{2} - \frac{d - 3}{2}, \dots, \frac{n}{2}, \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, \frac{n}{2} + \frac{d + 1}{2} \right\},$$

$$\min_{j \in V_d(1)} D(1, j) = \frac{n}{2} - \frac{d - 1}{2} = \frac{n - d + 1}{2},$$

$$\max_{j \in V_d(1)} D(1, j) = \frac{n}{2}.$$

The number of bad edges in G that include vertex 1 is

$$\begin{aligned} 0 & \text{ if } \alpha \leq (n - d + 1)/2 - 1 = (n - d - 1)/2, \\ 2 & \text{ if } \alpha = (n - d + 1)/2, \\ 4 & \text{ if } \alpha = (n - d + 3)/2, \\ & \vdots \\ d - 1 & \text{ if } \alpha = n/2 - 1 = [n - d + (d - 2)]/2, \\ d & \text{ if } \alpha \geq n/2. \end{aligned}$$

By symmetry, the same thing holds for every other vertex. Because each edge involves two vertices, we have $m(n, \alpha, d) = 0$ if $\alpha \leq (n - d - 1)/2, m(n, \alpha, d) = nd/2$ if $\alpha \geq n/2$, and

$$m(n, \alpha, d) = \frac{n(j + 1)}{2} \text{ if } \alpha = \frac{n - d + j}{2} \text{ for } j = 1, 3, \dots, d - 2.$$

Suppose d is even and n is odd. Then

$$V_d(1) = \left\{ \frac{n - d + 3}{2}, \dots, \frac{n + 1}{2}, \frac{n + 3}{2}, \dots, \frac{n + d + 1}{2} \right\},$$

$$\begin{aligned} \min_{j \in V_d(1)} D(1, j) &= \frac{n-d+1}{2}, \\ \max_{j \in V_d(1)} D(1, j) &= \frac{n-1}{2}. \end{aligned}$$

The number of bad edges in G that include vertex 1 is

$$\begin{aligned} 0 & \text{ if } \alpha \leq (n-d-1)/2, \\ 2 & \text{ if } \alpha = (n-d+1)/2, \\ 4 & \text{ if } \alpha = (n-d+3)/2, \\ & \vdots \\ d-2 & \text{ if } \alpha = (n-3)/2 = [n-d+(d-3)]/2, \\ d & \text{ if } \alpha \geq (n-1)/2. \end{aligned}$$

It follows that $m(n, \alpha, d) = 0$ if $\alpha \leq (n-d-1)/2$, $m(n, \alpha, d) = nd/2$ if $\alpha \geq \lfloor n/2 \rfloor$, and

$$m(n, \alpha, d) = \frac{n(j+1)}{2} \text{ if } \alpha = \frac{n-d+j}{2} \text{ for } j = 1, 3, \dots, d-3.$$

Suppose henceforth that d and n are even. The proof of Lemma 3.1(a) in [1] defines a connected G for which $m(n, \alpha, d) = 0$ whenever $\alpha < (n-d)/2$, or $\alpha \leq (n-d-1)/2$, so assume henceforth that $\alpha \geq (n-d)/2$. The $d-1$ vertices j for which $D(i, j)$ is as large as possible are those in $V_{d-1}(i)$, so we take $\{i, j\} \in E$ whenever $j \in V_{d-1}(i)$. We need one more edge for each vertex to attain d -regularity, and for simplicity take $\{1, 2\}, \{3, 4\}, \dots, \{n-1, n\}$ in E . To see that this will not overcount m , observe that

$$\begin{aligned} V_{d-1}(1) &= \left\{ \frac{n}{2} - \frac{d-4}{2}, \dots, \frac{n}{2}, \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, \frac{n}{2} + \frac{d}{2} \right\}, \\ \min_{j \in V_{d-1}(1)} D(1, j) &= \frac{n-d}{2} + 1, \\ \max_{j \in V_{d-1}(1)} D(1, j) &= \frac{n}{2}. \end{aligned}$$

Consequently, when $\alpha \geq (n-d)/2$, the final edge for vertex 1 must be a bad edge, so nothing is lost by taking it as $\{1, 2\}$. It follows that the number of bad edges in G that include vertex 1 is

$$\begin{aligned} 1 & \text{ if } \alpha = (n-d)/2, \\ 3 & \text{ if } \alpha = (n-d+2)/2, \\ 5 & \text{ if } \alpha = (n-d+4)/2, \\ & \vdots \\ d-1 & \text{ if } \alpha = n/2 - 1 = [n-d+(d-2)]/2, \\ d & \text{ if } \alpha \geq n/2. \end{aligned}$$

Because the same thing holds for every other vertex, we have $m(n, \alpha, d) = nd/2$ if $\alpha \geq n/2$ along with

$$m(n, \alpha, d) = \frac{n(j+1)}{2} \text{ if } \alpha = \frac{n-d+j}{2} \text{ for } j = 0, 2, 4, \dots, d-2.$$

3. Copies of a complete graph. Assume for Theorem 1.3 that $d \geq 3$ and $G = kK_{d+1}$ for some positive integer k with $n = k(d + 1)$.

If $\alpha \geq \lfloor n/2 \rfloor$, then, regardless of f , all edges are bad and $N(k, \alpha, d) = nd/2 = kd(d + 1)/2$. At the other extreme, where $\alpha \leq k - 1$, assign the color set $\{j, k + j, 2k + j, \dots, dk + j\}$ to the vertices of the j th copy of K_{d+1} ($j = 1, 2, \dots, k$) to conclude that $N(k, \alpha d) = 0$ when $\alpha \leq k - 1$. It is easily seen that if $\alpha \geq k$, then every copy of K_{d+1} has at least one bad edge regardless of f , and, if $\alpha < \lfloor n/2 \rfloor$, then not all edges are bad for some assignments. This concludes the proof of the first part of Theorem 1.3.

Assume henceforth that $t \in \{1, 2, \dots\}$, $d + 1 \geq 2t$, and that α is a positive integer that satisfies

$$\left\lfloor \frac{(d + 1)(k - 1)}{d + 2 - t} \right\rfloor + 1 \leq \alpha \leq \left\lfloor \frac{(d + 1)(k - 1)}{d + 1 - t} \right\rfloor.$$

The lower bound with $t \geq 1$ implies that $\alpha \geq k$.

LEMMA 3.1. *Every copy of K_{d+1} in G has at least t bad edges for every assignment. There are assignments for which at least one K_{d+1} in G has exactly t bad edges.*

Proof. Consider a coloring of the vertices of one copy of K_{d+1} with the following colors for which the first t pairs have absolute differences of 1:

$$1, 2, \alpha + 3, \alpha + 4, 2\alpha + 5, 2\alpha + 6, \dots, (t - 1)\alpha + 2t - 1, (t - 1)\alpha + 2t, \\ t\alpha + 2t + 1, (t + 1)\alpha + 2t + 2, \dots, (d - t)\alpha + d + 1.$$

The absolute difference between adjacent terms is $\alpha + 1$ if it is not 1. Consequently, if $(n + 1) - [(d - t)\alpha + d + 1] \geq \alpha + 1$, for the first and last terms, then this K_{d+1} has exactly t bad edges. The preceding inequality can be rewritten as $\alpha(d + 1 - t) \leq (d + 1)(k - 1)$, which is tantamount to the upper bound on α stated just prior to the lemma.

Contrary to the first part of the lemma, suppose an assignment f has $t - 1$ or fewer bad edges in some K_{d+1} . Without loss of generality, let f 's colors for this K_{d+1} be

$$1 = c_1 < c_2 < c_3 < \dots < c_{d+1} = (n + 1) - (\alpha + 1) = k(d + 1) - \alpha$$

so that c_{d+1} is as large as possible without interfering with c_1 . We then require $c_{i+1} - c_i \geq \alpha + 1$ for at least $(d + 1) - (t - 1) - 1 = d + 1 - t$ values of $i \in \{1, 2, \dots, d\}$ and $c_{i+1} - c_i \geq 1$ for the others. Therefore

$$k(d + 1) - \alpha - 1 = c_{d+1} - c_1 = \sum_{i=1}^d (c_{i+1} - c_i) \geq (d + 1 - t)(\alpha + 1) + t - 1,$$

which simplifies to $\alpha \leq (d + 1)(k - 1)/(d + 2 - t)$. This inequality contradicts the lower bound on α stated just prior to the lemma, so we conclude that every f has at least t bad edges in every K_{d+1} in G . \square

To complete the proof of Theorem 1.3, we need to prove that there is an assignment for which every K_{d+1} in G has exactly t bad edges, so $N(k, \alpha, d) \leq tk$. We do this for α when it equals its upper bound, i.e., for

$$\alpha = \left\lfloor \frac{(d + 1)(k - 1)}{d + 1 - t} \right\rfloor.$$

Any assignment that achieves the desired result for this α serves also for smaller α in the interval noted just prior to Lemma 3.1. According to the lemma, every assignment for $G = kK_{d+1}$ has at least tk bad edges, and therefore $N(k, \alpha, d) = tk$.

Several new parameters are used to define an assignment that gives every K_{d+1} exactly t bad edges. The first few are p , β , and γ defined by

$$\begin{aligned} p &= d + 1 - t && (\text{so } p \geq t \text{ because } d + 1 \geq 2t), \\ p &= \beta(k - 1) + \gamma && \text{with } 0 \leq \gamma < k - 1, \end{aligned}$$

so $\beta = \lfloor p/(k - 1) \rfloor$ and $\gamma = p - \beta(k - 1)$. Note that the bounds on α require $k - 1 > 0$ so that the range for α is not empty.

We begin our construction that leads to the desired assignment by defining C as a clockwise circular arrangement of p successive copies of $12 \dots k$, beginning from a designated starting point as shown on the top left of Figure 3.1. We refer to each $12 \dots k$ copy as a *long block* and number them consecutively as $1, 2, \dots, p$ from the start. The first $k - 1$ long blocks, the next $k - 1$ long blocks, and so forth each comprise a *superblock*, so C has β superblocks. The final γ long blocks before we arrive back at the start are the *remainder long blocks* or the *remainder region*. This is illustrated on the top right of Figure 3.1 for $p = 8$, $k = 4$, $\beta = 2$, and $\gamma = 2$. By definition, $d - t = p - 1 = 7$. Because $d + 1 \geq 2t$, we require $t \leq 8$. We also require the α interval prior to Lemma 3.1 to be nonempty, which it is for $t \in \{3, 6, 8\}$ but is not for other $t \leq 8$. Further illustrations for Figure 3.1 for the parameters used on the top right assume that $t = 3$ with $d = 10$ and $\alpha = 4$.

We now define short blocks for C and explain how an assignment for (k, α, d) will be obtained from C . A *short block* is any contiguous sequence of $k - 1$ terms of C . A short block that consists of the first $k - 1$ terms of C (clockwise from the start), or the second $k - 1$ terms of C , and so forth is a *main short block*, abbreviated as MSB. Because $(k - 1)k = k(k - 1)$, each superblock ($k - 1$ long blocks of k terms each) contains precisely k MSBs. The β superblocks therefore encompass βk MSBs. The γ remainder long blocks from the end of the final superblock around to the start have $\lfloor \gamma k / (k - 1) \rfloor$ MSBs plus 0 to $k - 2$ final terms of C before we arrive back at the start. The middle left diagram of Figure 3.1 pictures the MSBs and the final two terms of C for the configuration in the top right of the figure.

The next two parameters involved in the construction are

$$q = \left\lfloor \frac{t(k - 1)}{p} \right\rfloor = \alpha - (k - 1)$$

and

$$u = (k - 1) - q,$$

so $q + u = k - 1$. Because $\alpha \geq k$ and $p \geq t$, $1 \leq q \leq k - 1$. Our objective is to identify tk terms of C , referred to as *circled terms*, so that

- (i) every short block has at least q circled terms,
- (ii) every $j \in \{1, 2, \dots, k\}$ is circled exactly t times.

Given (i) and (ii), an assignment for $G = kK_{d+1}$ is formed as follows. Beginning at

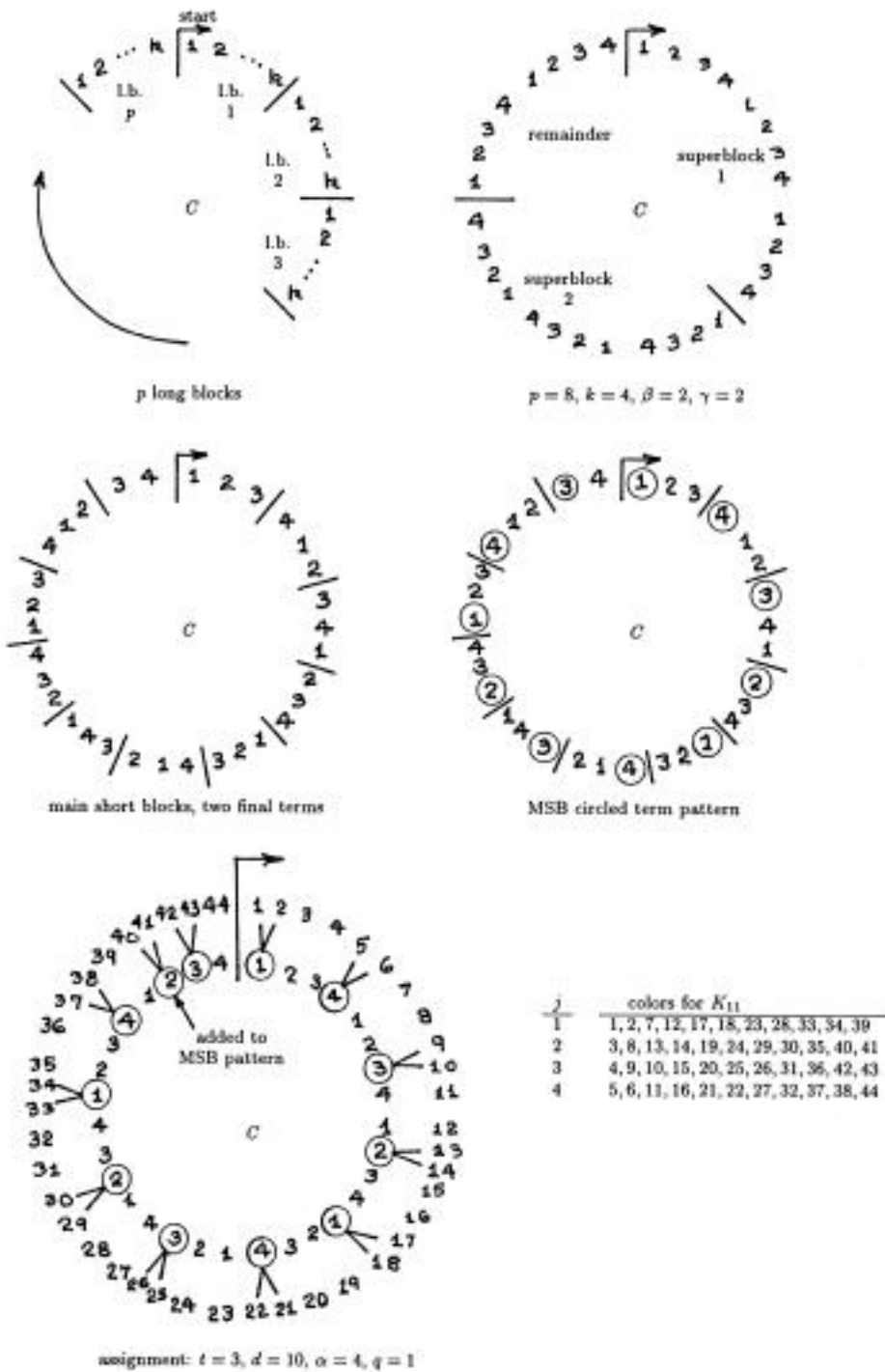


FIG. 3.1.

the start and proceeding clockwise around C , assign colors $1, 2, \dots$ successively to the terms of C so that every circled term is assigned two adjacent integers and every term that is not circled is assigned a single integer. Because C has $pk = (d+1-t)k$ terms of which tk are circled, the largest color used is $(d+1-t)k + tk = k(d+1) = n$. The colors assigned to the j th copy of K_{d+1} in G are those assigned to the terms of C labeled j for $j = 1, 2, \dots, k$. Thus each K_{d+1} receives $d+1$ colors, among which, by (ii), there are t pairs with absolute difference 1 so that K_{d+1} has at least t bad edges. However, that is all. Between consecutive occurrences of j around C there are $k-1$ other terms of C and, by (i), at least q of these $k-1$ are circled, so there is an interval of at least $k-1+q = \alpha$ assigned colors between the j terms. It follows that the distance between j -term colors that is not 1 is at least $\alpha+1$. Hence every copy of K_{d+1} in G has exactly t bad edges.

We illustrate the assignment procedure at the bottom of Figure 3.1 for $p = 8$, $k = 4$, $t = 3$, $d = 10$, and $\alpha = 4$. The circled terms are easily seen to satisfy (i) and (ii): every short block of three terms has at least $q = 4 - 3 = 1$ circled terms, and every j is circled exactly three times. The colors assigned to the vertices of the four copies of K_{11} are summarized at the lower right of the figure.

Conditions (i) and (ii) will be satisfied by a primary step and a secondary step. The primary step defines a *circled term pattern* P which identifies the q of the $k-1$ terms that are circled (going clockwise) in each MSB. The pattern is the same for every MSB and is continued into the final terms between the end of the last MSB and the start. For $q = 1$ and $k-1 = 3$ in Figure 3.1, P specifies the first of the three terms in each MSB as the one to be circled. This is shown on the middle right diagram with continuation into the final two terms where the first 3 is circled.

We will define P in such a way that (i) holds and no j in C is circled more than t times. The primary step will usually circle most j 's t times, but some may receive fewer than t circles. The secondary step then adds circles for the latter cases so that (ii) holds along with (i). When a circle for j is added in the secondary step, any j terms as yet uncircled can receive the new circle. On the middle right of Figure 3.1, each $j \in \{1, 3, 4\}$ has its full complement of $t = 3$ circles from the primary step, but $j = 2$ has only two circles. We therefore circle one more $j = 2$ in the lower left diagram. Although the particular choice of the new circle obviously affects the assignment, it does not affect the fact that each copy of K_{d+1} has exactly t bad edges.

In what follows, we suppress the secondary step because its only role is to add circles to satisfy (ii) and therefore provide a full assignment. The important task is to define P so that every short block has at least q circled terms (which is automatic by an obvious shift argument unless the short block overlaps the starting position) and no j is circled more than t times. There are cases in which any circled term pattern suffices for the desired result, whereas other cases require great care in defining a suitable P . The following lemma identifies one case in which the particular structure of P is immaterial.

LEMMA 3.2. *With $p = \beta(k-1) + \gamma$, every circled term pattern P has the property that every $j \in \{1, 2, \dots, k\}$ is circled exactly βq times in the β superblocks of C . If $\gamma = 0$, then, for every P , every short block has at least q circled terms and no $j \in \{1, 2, \dots, k\}$ is circled more than t times.*

Proof. Every superblock consists of k MSBs and therefore has qk circled terms. Each of the k MSBs in a superblock begins with a different j , and because the same P is used in every MSB the qk circled terms in a superblock are evenly divided among the k values of j . Hence every j is circled βq times in the β superblocks.

Suppose $\gamma = 0$, so there is no remainder. Every MSB has precisely q circled terms, so successive one-term shifts and the use of the same P in every MSB imply that every short block has precisely q circled terms. Moreover,

$$q = \left\lfloor \frac{t(k-1)}{\beta(k-1)} \right\rfloor \leq \frac{t}{\beta},$$

so $\beta q \leq t$. \square

Another straightforward case arises when $q = k - 1$ and $u = 0$, in which event P must circle every term in C . An example occurs for $t = 8$ and $d = 15$ for the top right diagram of Figure 3.1. Then the lower and upper bounds on α prior to Lemma 3.1 are both 6, so $\alpha = 6$ and $q = \alpha - (k - 1) = 3 = k - 1$.

LEMMA 3.3. *Suppose $q = k - 1$. Then $p = t$, every term of C is circled, and each $j \in \{1, 2, \dots, k\}$ is circled exactly t times.*

Proof. Suppose $q = k - 1$. Because every term of C is circled and C has pk terms, every j is circled exactly p times. Moreover, $\alpha = q + k - 1 = 2(k - 1)$, and hence

$$2(k - 1) = \left\lfloor \frac{(d + 1)(k - 1)}{d + 1 - t} \right\rfloor.$$

This implies that $(d + 1)/(d + 1 - t) \geq 2$, i.e., $2t \geq d + 1$. However, $d + 1 \geq 2t$ by hypothesis, so $d + 1 = 2t$. It follows that $t = d + 1 - t = p$. \square

It follows from Lemmas 3.2 and 3.3 along with our discussion of (i) and (ii) that the latter part of Theorem 1.3 holds if either $\gamma = 0$ or $u = 0$, so assume henceforth that q, u , and γ are positive. Then

$$p = \beta(k - 1) + \gamma \quad \text{with} \quad 0 < \gamma < k - 1,$$

and, by Lemma 3.2, every $j \in \{1, 2, \dots, k\}$ is circled exactly βq times in the β superblocks of C that precede the γ remainder long blocks. For every circled term pattern P let $r(P)$ denote the maximum number of times a $j \in \{1, 2, \dots, k\}$ is circled in the γ remainder long blocks when P is used for the circling operation. Then every $j \in \{1, 2, \dots, k\}$ is circled at most t times in C if and only if

$$\beta q + r(P) \leq t.$$

LEMMA 3.4. *Every $j \in \{1, 2, \dots, k\}$ is circled at most t times in C when P is used if*

$$\left\lceil \frac{q\gamma}{k - 1} \right\rceil \geq r(P).$$

Proof. Because $q = \lfloor t(k - 1)/[\beta(k - 1) + \gamma] \rfloor$, we have

$$\frac{t(k - 1)}{\beta(k - 1) + \gamma} \geq q$$

and therefore have $t \geq \beta q + q\gamma/(k - 1)$. Hence $t \geq \beta q + \lceil q\gamma/(k - 1) \rceil$. It follows that $t \geq \beta q + r(P)$ if the inequality of Lemma 3.4 holds. \square

We proceed to define P so that the inequality of Lemma 3.4 holds and the short blocks that overlap the starting position of C have at least q circled terms. This is

the final step in the proof of Theorem 1.3. We begin it by assuming that $q \geq u > 0$ and then consider $u > q > 0$.

Given $q \geq u > 0$, let a and b satisfy

$$q = au + b, \quad a \geq 1, \quad 0 \leq b < u.$$

Then $k - 1 = q + u = (a + 1)u + b$.

Suppose $b = 0$. Then $q = au$ and $k - 1 = (a + 1)u$. Let $x = x_1x_2 \cdots x_{k-1}$ denote the $k - 1$ successive terms of an MSB. Define P as the pattern that circles every x_i except those for which $i \in \{a + 1, 2(a + 1), \dots, u(a + 1)\}$. Thus, when x is divided into u runs of $a + 1$ terms each, P circles the first a terms in each run. As noted before, P 's pattern of circles is continued into the final terms of C beyond the last MSB. Thus C either ends with a run of circled terms or concludes with an uncircled k preceded by a circled terms. Because the first MSB after the start has pattern P , every short block in C has at least q circled terms.

Moreover, because the γ remainder long blocks have γ instances of each $j \in \{1, 2, \dots, k\}$ and because P begins with circled terms and the first MSB in the γ region begins with 1, $j = 1$ has as many circled instances in that region as any other j . Thus $r(P)$ is the number of circled 1's in the γ region. Now observe that the γ remainder long blocks with γk terms have exactly $\gamma = \lfloor \gamma k / (k - 1) \rfloor$ complete MSBs, for if $\lfloor \gamma k / (k - 1) \rfloor \geq \gamma + 1$, then $\gamma k / (k - 1) \geq \gamma + 1$ and therefore $\gamma \geq k - 1$, contradicting $\gamma < k - 1$. This in turn implies that there are $\gamma = \gamma k - \gamma(k - 1)$ final terms before the start, namely $k - \gamma + 1, k - \gamma + 2, \dots, k$ with $k - \gamma + 1 \geq 2$. Because the terms in the MSBs in the γ region are $12 \dots k - 1$ for the first, $k12 \dots k - 2$ for the second, and so forth, it follows that the number of circled 1's in the γ region is the number of circled terms in the first γ positions of an MSB, i.e., the number of circled x_i in $x_1x_2 \cdots x_\gamma$. This number is $\gamma - \lfloor \gamma / (a + 1) \rfloor$, and therefore

$$r(P) = \gamma - \lfloor \gamma / (a + 1) \rfloor.$$

Hence, with $q / (k - 1) = a / (a + 1)$, the inequality of Lemma 3.4 holds if and only if

$$\left\lceil \frac{a\gamma}{a + 1} \right\rceil \geq \gamma - \left\lfloor \frac{\gamma}{a + 1} \right\rfloor.$$

This is true because

$$\left\lceil \frac{a\gamma}{a + 1} \right\rceil = \left\lceil \gamma - \frac{\gamma}{a + 1} \right\rceil = \gamma + \left\lceil -\frac{\gamma}{a + 1} \right\rceil = \gamma - \left\lfloor \frac{\gamma}{a + 1} \right\rfloor.$$

This completes the proof for $q = au$.

Continuing with $q = au + b$, $a \geq 1$, and $0 \leq b < u$, suppose that $b \geq 1$. When $b = 0$, P had u runs of a circled terms, each followed by an uncircled term. With the circled-term runs numbered $1, 2, \dots, u$, left to right, we use the same basic P for $b \geq 1$ except that the runs whose numbers are in

$$\left\{ \left\lfloor 1 + \frac{iu}{b} \right\rfloor : i = 0, 1, \dots, b - 1 \right\}$$

are increased from a circled terms to $a + 1$ circled terms. This gives $q = au + b$ circled terms and, as before, u uncircled terms, for a total of $k - 1 = q + u$ terms in the MSB. To illustrate, suppose $u = 10$ and $b = 3$. Then the first, fourth, and seventh

circled-term runs in P have $(a + 1)$ terms apiece, while runs 2, 3, 5, 6, 8, 9, and 10 have a terms apiece. If $u = 10$ and $b = 7$, the $(a + 1)$ -term runs are numbered 1, 2, 3, 5, 6, 8, and 9, and the a -term runs are numbered 4, 7, and 10. These two cases have the following *run patterns* when 1 denotes an $(a + 1)$ -term run and 0 denotes an a -term run:

$$\begin{aligned} (u, b) = (10, 3) : & \quad 1001001000, \\ (u, b) = (10, 7) : & \quad 1110110110. \end{aligned}$$

Let $P(u, b)$ denote the circled-term pattern P defined in the preceding paragraph for $1 \leq b \leq u - 1$, and let

$$e_i = \left\lfloor 1 + \frac{(i - 1)u}{b} \right\rfloor \quad \text{for } i = 1, \dots, b,$$

so the circled-term runs with $a + 1$ terms are numbered $1 = e_1, e_2, \dots, e_b$. We have defined $P(u, b)$ so that it spreads the uncircled one-term runs fairly evenly around C , begins every MSB with a run of $(a + 1)$ circled terms, ends every MSB with an uncircled term, and shows a slight preference for the longer circled-term runs near the beginning of the MSB. The following lemma notes a remarkable feature of run patterns as defined at the end of the preceding paragraph. It is illustrated by the run patterns for $(u, b) = (10, 3)$ and $(u, b) = (10, 7)$.

LEMMA 3.5. *The run pattern for $(u, u - b)$ is obtained from the run pattern for (u, b) by changing all 1's to 0's and all 0's to 1's in the (u, b) run pattern and then reversing the entire sequence.*

Remark. With $a \geq 1$ and $1 \leq b \leq u - 1$, the run pattern for (u, b) refers to $P(u, b)$ with $q = au + b$, and the run pattern for $(u, u - b)$ refers to $P(u, u - b)$ with $q' = au + (u - b)$.

Proof. The 1's in the run pattern for (u, b) are the terms in that pattern numbered $e_i = \lfloor 1 + (i - 1)u/b \rfloor$ for $i = 1, \dots, b$. The 1's in the run pattern for $(u, u - b)$ are the terms in that pattern numbered

$$f_j = \left\lfloor 1 + \frac{(j - 1)u}{u - b} \right\rfloor \quad \text{for } j = 1, \dots, u - b.$$

Let $g_j = u + 1 - f_j$ so that

$$\begin{aligned} g_j &= u + 1 - \left\lfloor 1 + \frac{(j - 1)u}{u - b} \right\rfloor = u + 1 + \left\lceil -1 - \frac{(j - 1)u}{u - b} \right\rceil \\ &= \left\lfloor u - \frac{(j - 1)u}{u - b} \right\rfloor \quad \text{for } j = 1, \dots, u - b. \end{aligned}$$

The operations described in the lemma for going from the run pattern for (u, b) to the run pattern for $(u, u - b)$ are tantamount to the assertion that

$$\{e_1, e_2, \dots, e_b\} \cap \{g_1, g_2, \dots, g_{u-b}\} = \emptyset.$$

Suppose to the contrary that $e_i = g_j$ for some i and j so that

$$\left\lfloor 1 + \frac{(i - 1)u}{b} \right\rfloor = \left\lfloor u - \frac{(j - 1)u}{u - b} \right\rfloor = c$$

for a positive integer c . Then

$$c \leq 1 + \frac{(i-1)u}{b} < c + 1$$

and

$$c - 1 < u - \frac{(j-1)u}{u-b} \leq c.$$

The first of these double inequalities can be written as

$$\frac{b(c-1)}{u} \leq i-1 < \frac{bc}{u}$$

and, with $z = j-1-u+b+c$, the second can be written as

$$\frac{bc}{u} \leq z < \frac{b(c-1)}{u} + 1.$$

These imply that $i-1 < z < i$, which is impossible because i and z are integers. \square

We now use Lemma 3.5 to prove a result that reflects the propensity of $P(u, b)$ to position longer circled-term runs near the beginning of its pattern. Until further notice, *run* denotes *circled-term run* with either a or $a+1$ circled terms, so $P(u, b)$ has b runs of length $a+1$ and $u-b$ runs of length a . Two such runs are contiguous if they are separated by a single uncircled term.

LEMMA 3.6. *With $q = au + b$, $a \geq 1$ and $1 \leq b \leq u - 1$, suppose $1 \leq s \leq u - 1$. Then the number of circled terms in s successively contiguous runs of $P(u, b)$ is maximized by the first s runs and minimized by the final s runs.*

Proof. Let $w = u/b > 1$, so $e_1 = [1 + 0w]$, $e_2 = [1 + w]$, \dots , $e_b = [1 + (b-1)w]$ with w as the constant difference between the floor arguments of e_{i+1} and e_i . Given $1 \leq s \leq u - 1$, the lemma claims that $[1, s]$ contains as many e_i 's as each of $[2, s+1]$, $[3, s+2]$, \dots , $[u-s+1, u]$, and that $[u-s+1, u]$ contains as few e_i 's as each of $[1, s]$, \dots , $[u-s, u-1]$. The max claim for $[1, s]$ follows immediately from the fact that the floor argument of e_1 is 1.

The min claim for $[u-s+1, u]$ is implied by Lemma 3.5 and the max claim for $[1, s]$. According to Lemma 3.5, a *right-to-left* traversal through the run pattern of $P(u, b)$ is the same as a *left-to-right* traversal through the run pattern of $P(u, u-b)$ with 0's and 1's interchanged. The min claim for $[u-s+1, u]$ with respect to $P(u, b)$ is therefore identical to the max claim for $[1, s]$ with respect to $P(u, u-b)$, and, since the latter is true, so is the former. \square

Our next lemma notes two key implications of the construction of $P(u, b)$. Recall that $r(P)$ is the maximum number of times a $j \in \{1, 2, \dots, k\}$ is circled in the γ remainder long blocks when pattern P is used for the circling operation.

LEMMA 3.7. *Suppose $q = au + b$, $a \geq 1$, $1 \leq b \leq u - 1$, and $\gamma > 0$. Then, with terms of C circled according to $P(u, b)$, every short block in C has at least q circled terms and $r(P(u, b))$ is the number of times $j = 1$ is circled in the remainder region.*

Proof. To prove the short block assertion, let AB denote a short block that overlaps the start of C with A left of and B right of the start. To minimize the number of circled terms in AB , we can do no better than to have A 's final term (next to the start) uncircled. Going counterclockwise from there, it follows from the final part of Lemma 3.6 that the number of circled terms in A will be minimized when A

is the right end of an MSB with pattern $P(u, b)$. When A is transposed clockwise immediately to the right of B , BA has the structure of an MSB circled according to $P(u, b)$, and therefore it, and AB , has q circled terms.

To prove the assertion for r , let $2P(u, b)$ denote two adjacent copies of the $P(u, b)$ pattern with $2(k - 1)$ positions from start to finish. Also let $h(j)$ be the number of times j is circled in the remainder region. Then, for every $j \in \{1, 2, \dots, k\}$, $h(j)$ is the number of circled terms of $2P(u, b)$ in positions j through $j + \gamma - 1$. This follows from the fact noted earlier for $b = 0$ after Lemma 3.4 that the remainder region has exactly γ complete MSBs followed by γ final terms, namely $k - \gamma + 1$ through k , just before the start. For example, the remainder region terms for $k = 7$ and $\gamma = 3$ are

start

1 2 3 4 5 6 | 7 1 2 3 4 5 | 6 7 1 2 3 4 | 5 6 7 |.

By definition, $h(k) = h(1)$. Contrary to the final conclusion of Lemma 3.7, suppose $h(j') > h(1)$ for some $j' \in \{2, \dots, k - 1\}$. Then the position's interval $[1, \gamma]$ of $2P(u, b)$ has more *uncircled* terms than $[j', j' + \gamma - 1]$. Translate the latter interval leftward to $[j, j + \gamma - 1]$ until the term that precedes it is uncircled, i.e., so that it begins with a run. This does not increase its uncircled terms, so $[1, \gamma]$ has more uncircled terms than $[j, j + \gamma - 1]$, and both begin with runs.

Suppose $j + \gamma - 1 < k - 1$. Extend $[j, j + \gamma - 1]$ rightward as far as possible without increasing its uncircled terms. (The final term of $P(u, b)$ is uncircled.) Also remove the right end of $[1, \gamma]$ so that it has the same number of uncircled terms as $[j, j + \gamma - 1]$ and ends just before an uncircled term. The modified $[1, \gamma]$ and $[j, j + \gamma - 1]$ then contain the same number of runs, and the latter has more circled terms than the former, which contradicts the max part of Lemma 3.6.

Suppose $j + \gamma - 1 = k - 1$, so the last term of $[j, j + \gamma - 1]$ is uncircled and has a run just before that term. Then $[1, \gamma]$ has more runs than $[j, j + \gamma - 1]$, but fewer circled terms, again contradicting the max part of Lemma 3.6.

Finally, suppose $j + \gamma - 1 \geq k$, so $[j, j + \gamma - 1]$ goes into the second $P(u, b)$. Subtract the same-patterned initial segment of $[1, \gamma]$ and the final segment of $[j, j + \gamma - 1]$ so that

$$\begin{aligned} [1, \gamma] &\rightarrow [\gamma + j + 1 - k, \gamma], \\ [j, j + \gamma - 1] &\rightarrow [j, k - 1], \end{aligned}$$

each with $k - j$ terms. Because the removed segments have the same pattern, $[\gamma + j + 1 - k, \gamma]$ has more uncircled terms than $[j, k - 1]$, so $[j, k - 1]$ has more circled terms. Suppose $[j, k - 1]$ has s uncircled terms, one of which is $k - 1$, and therefore has s runs. Then, even if $[\gamma + j + 1 - k, \gamma]$ begins and ends with uncircled terms, it contains at least s runs. However, it also has fewer circled terms than $[j, k - 1]$, and this contradicts the min part of Lemma 3.6. \square

We are now in position to complete the proof of the latter part of Theorem 1.3 when $\lfloor t(k - 1)/(d + 1 - t) \rfloor \geq (k - 1)/2$, i.e., when $q \geq u$.

LEMMA 3.8. *The latter part of Theorem 1.3 is true if $q \geq u$.*

Proof. According to Lemmas 3.4 and 3.7 and supporting discussion, it suffices to prove that

$$(3.1) \quad \left\lceil \frac{q\gamma}{k - 1} \right\rceil \geq r(P(u, b))$$

when $q = au + b$, $a \geq 1$, $1 \leq b \leq u - 1$, and $\gamma > 0$, where $r(P(u, b))$ is the number of times $j = 1$ is circled in the remainder region or, equivalently, the number of circled terms in the first γ terms of $P(u, b)$.

The number of *circled* terms of $P(u, b)$ through the final position of its s th run is

$$z_s = as + |\{i : e_i \leq s\}|,$$

and the number of terms of $P(u, b)$ to the same point is $z_s + s - 1$. Because $q = au + b$ and $k - 1 = (a + 1)u + b$, (3.1) holds when $\gamma = z_s + s - 1$ if and only if

$$(3.2) \quad \left\lceil \frac{(au + b)(z_s + s - 1)}{(a + 1)u + b} \right\rceil \geq z_s.$$

If (3.2) is true and γ is increased by 1 to include the next uncircled term, it remains true because the ceiling argument increases and the right side remains at z_s . If we go the other way and reduce γ by h terms while staying within the s th run, then (3.1) is tantamount to (3.2) when z_s on both sides is replaced by $z_s - h$, and it is easily seen that this version of (3.1) holds when (3.2) holds as stated.

Consequently, we need only show that (3.2) holds for $s = 1, 2, \dots, u$.

The ceiling argument of (3.2) equals

$$z_s + \frac{(au + b)(s - 1) - z_s u}{(a + 1)u + b},$$

and therefore (3.2) is true if and only if

$$\frac{(au + b)(s - 1) - z_s u}{(a + 1)u + b} > -1.$$

This inequality reduces to $z_s u < u(as + 1) + bs$ which, after division by u and substitution of $as + |\{i : e_i \leq s\}|$ for z_s , can be rewritten as

$$|\{i : 1 \leq e_i \leq s\}| < 1 + \frac{bs}{u}.$$

Now

$$\begin{aligned} e_i \leq s &\Leftrightarrow \left\lceil 1 + \frac{(i - 1)u}{b} \right\rceil \leq s \\ &\Leftrightarrow 1 + \frac{(i - 1)u}{b} < s + 1 \\ &\Leftrightarrow i < 1 + \frac{bs}{u}, \end{aligned}$$

so the preceding inequality is the same as

$$\left| \left\{ i : 1 \leq i < 1 + \frac{bs}{u} \right\} \right| < 1 + \frac{bs}{u}.$$

The left side here is bs/u if bs/u is an integer, and is $1 + \lfloor bs/u \rfloor$ otherwise, so the inequality holds for $s = 1, 2, \dots, u$. Consequently, (3.2) holds in all cases and the proof is complete. \square

It remains to show that the latter part of Theorem 1.3 holds when $u > q$. The proof for this case is dually similar to the proof for $q > u$, and we sketch only its main

points. The fundamental change from the preceding proof, which is suggested by the analysis for Lemma 3.5, is to interchange “circled” and “uncircled” and construct $P(q, b)$ right-to-left in the left-to-right manner of $P(u, b)$ for $q > u$. With

$$u = aq + b, \quad a \geq 1, \quad 0 \leq b \leq q - 1,$$

we now refer to a *run* as a sequence of a or $a + 1$ *uncircled* terms in P without intervention of a circled term. When $b = 0$, P begins from the right with a run of a terms followed by a circled term and continues this pattern q times to the left end. When $b \geq 1$, $P(q, b)$ is like P for $b = 0$ except that each run whose run number right-to-left is in

$$\left\{ \left\lfloor 1 + \frac{iq}{b} \right\rfloor : i = 0, 1, \dots, b - 1 \right\}$$

is increased from a to $a + 1$ uncircled terms.

Despite the dual construction of $P(q, b)$, it is used in the original sense to form an assignment, as in Figure 3.1. Going left-to-right, $P(q, b)$ begins with a circled term, ends with an uncircled term, and has a slight propensity to position the circled terms near the beginning.

The analysis through the proof of Lemma 3.4 is not affected by $u > q$. Lemma 3.5 applies with the new meaning for “run” when q replaces u . Lemma 3.6 applies under the same changes for the right-to-left orientation. Going *left-to-right*, the new Lemma 3.6 says that the number of uncircled terms in s successively continuous runs of $P(q, b)$ is minimized by the first s runs and maximized by the final s runs. Lemma 3.7 applies when q and u are interchanged. In particular, $r(P(q, b))$, the maximum number of times some j is circled in the remainder region, equals the number of times $j = 1$ is circled in that region. As before, this is the number of circled terms in the first γ positions left-to-right of $P(q, b)$.

The desired inequality of Lemma 3.4 can be proved from the fact, established dually for (3.1) in the former $q > u$ analysis, that the number $r^*(P(q, b))$ of *uncircled* terms in the first $k - 1 - \gamma$ terms of an MSB going *right-to-left* satisfies

$$\left\lceil \frac{u(k - 1 - \gamma)}{k - 1} \right\rceil \geq r^*(P(q, b)).$$

Then the first γ *left-to-right* positions in an MSB have at least $u - \lceil u(k - 1 - \gamma)/(k - 1) \rceil$ uncircled terms and therefore have no more than $\gamma - \{u - \lceil u(k - 1 - \gamma)/(k - 1) \rceil\}$ circled terms. We therefore have

$$r(P(q, b)) \leq \gamma - u + \left\lceil \frac{u(k - 1 - \gamma)}{k - 1} \right\rceil = \left\lceil \frac{(k - 1 - u)\gamma}{k - 1} \right\rceil = \left\lceil \frac{q\gamma}{k - 1} \right\rceil,$$

which is the desired inequality of Lemma 3.4.

4. More for $d = 3$. To prove Theorem 1.2 for $d = 3$, suppose first that $n = 4k$ with G composed of k copies of K_4 . The α ranges for $M^* \in \{0, 6k\}$ are implied by the first part of Theorem 1.3, and those for $M^* \in \{k, 2k\}$ are implied by $t \in \{1, 2\}$ in the latter part of Theorem 1.3. This leaves only $\alpha = 2k - 1$ where there are $2k$ color pairs without interference, namely $\{i, 2k + i\}$ for $i = 1, \dots, 2k$. All such pairs can be used as edges in the k copies of K_4 , so $M^*(4k, 2k - 1, 3) = 6k - 2k = 4k$.

Assume henceforth that $d = 3$, $k \geq 0$, $n = 4k + 6$, and G consists of k copies of K_4 and either H_1 or H_2 as in Figure 1.1. Because Theorem 1.2 stated the results

for $n = 4k + 2$ and $k \geq 1$, we rewrite them for $n = 4k + 6$ and $k \geq 0$ as follows: $M^*(4k + 6, \alpha, 3)$ equals

- A.** 0 if $\alpha \leq k$,
- B.** 2 if $(k, \alpha) \in \{(0, 1), (1, 2)\}$,
- C.** k if $k \geq 2, k + 1 \leq \alpha \leq \lceil \frac{4k+1}{3} \rceil$, and it is
false that $k \equiv 0 \pmod{3}$ and $\alpha = \lceil \frac{4k+1}{3} \rceil$,
- D.** $2k$ if $k \geq 3, k \equiv 0 \pmod{3}$ and $\alpha = \lceil \frac{4k+1}{3} \rceil$,
- E.** $2k + 2$ if $\lceil \frac{4k+4}{3} \rceil \leq \alpha \leq 2k + 1$,
- F.** $4k + 6$ if $\alpha = 2k + 2$,
- G.** $6k + 9$ if $\alpha \geq 2k + 3$.

We verify these in reverse order except that **C** and **D** are left for last.

G. If $\alpha \geq 2k + 3$, then all edges are bad, so $M^* = nd/2 = 6k + 9$. ■

F. When $\alpha = 2k + 2$, there are $2k + 3$ pairs without interference, namely $\{i, 2k + 3 + i\}$ for $i = 1, \dots, 2k + 3$. The k K_4 's can use $2k$ of these as edges, and H_1 or H_2 can use the other three, so $M^* = 6k + 9 - (2k + 3) = 4k + 6$. ■

E. When $\lceil (4k + 4)/3 \rceil \leq \alpha \leq 2k + 1$, every C_3 has at least one bad edge. For example, if the colors of a C_3 are $1, \alpha + 2$, and $2\alpha + 3$, then $D(1, 2\alpha + 3) \leq \alpha$. It follows that every K_4 has at least two bad edges and H_2 , which contains two disjoint C_3 's, has at least two bad edges. Each K_4 has exactly two bad edges when their color sets are $\{i, i + 1, 2k + 3 + i, 2k + 4 + i\}$ for $i = 1, 3, 5, \dots, 2k - 1$. The other six colors are $2k + 1, 2k + 2, 2k + 3, 4k + 4, 4k + 5$, and $4k + 6$. These six can be assigned to the vertices of H_1 or H_2 to give no more than two bad edges, so $M^* = 2k + 2$. ■

B. The adversary's better strategy is H_1 for both of $(n, \alpha) = (6, 1)$ and $(n, \alpha) = (10, 2)$. The first of these is shown in the middle of Figure 1.1, where it is easily seen that H_1 forces at least two bad edges. Suppose $(n, \alpha) = (10, 2)$. Then K_4 must have at least one bad edge. If H_2 is used, there is only one bad edge overall when K_4 's colors are $1, 4, 7, 10$ and the colors for H_2 , clockwise from its top left vertex, are $2, 6, 9, 3, 8, 5$. If H_1 is used and K_4 has only one bad edge, H_1 must have at least one bad edge. The assignment of $4, 5, 9, 10$ to K_4 and $1, 6, 2, 7, 3, 8$ clockwise to the vertices of H_1 shows that $M^* = 2$ for $(n, \alpha) = (10, 2)$. ■

A. We show that $M^* = 0$ when $\alpha = k$. If the adversary chooses H_1 , use the four colors in each column of

1	2		k
$k + 2$	$k + 3$		$2k + 1$
$2k + 3$	$2k + 4$...	$3k + 2$
$3k + 4$	$3k + 5$		$4k + 3$

to color the vertices of each K_4 , and color the vertices of H_1 clockwise as $k + 1, 4k + 4, 2k + 2, 4k + 5, 3k + 3, 4k + 6$. If the adversary chooses H_2 and $k = 1$, color K_4 with $1, 3, 5, 8$ and H_2 with $6, 9, 2, 7, 10, 4$, clockwise from its top left vertex. If the adversary chooses H_2 and $k \geq 2$, use the four colors in each column of

	2	4	5		$k + 1$
$k + 2$	$k + 3$	$k + 5$	$k + 6$...	$2k + 2$
$2k + 4$	$2k + 5$	$2k + 6$	$2k + 7$		$3k + 3$
$3k + 5$	$3k + 7$	$3k + 8$	$3k + 9$		$4k + 5$
$4k + 6$					

to color the vertices of each K_4 , and color the vertices of H_2 clockwise from its top left vertex with 1, $k + 4$, 3, $3k + 6$, $2k + 3$, $3k + 4$. None of these assignments has a bad edge when $\alpha = k$. ■

C and **D**. $M^* \geq k$ for **C**, and $M^* \geq 2k$ for **D** follow from the facts that (i) if $\alpha \geq k + 1$, then every K_4 has at least one bad edge (consider 1, $\alpha + 2$, $2\alpha + 3$, $3\alpha + 4$) and (ii) if $k = 3t$ with $t \geq 1$ and $\alpha = \lceil (4(3t) + 1)/3 \rceil = 4t + 1$, then every K_4 has at least two bad edges. For (ii), if every C_3 in a K_4 has a bad edge, then the K_4 has at least two bad edges, and if some C_3 in a K_4 has no bad edges, then the color pairs in this C_3 all have $D = \alpha + 1 = 4t + 2$ (consider 1, $\alpha + 2$, $2\alpha + 3$, i.e., 1, $4t + 3$, $8t + 5$, with $12t + 7 - (8t + 5) = 4t + 2 = \alpha + 1$) and any fourth color forces two bad edges.

We show next that $M^* = 2k$ for **D**. Let $k = 3t$, $t \geq 1$, with $\alpha = 4t + 1$. If H_1 is chosen, use the colors in each column of

$$\begin{array}{cccc} 1 & 3 & & 6t - 1 \\ 2 & 4 & \dots & 6t \\ 6t + 4 & 6t + 6 & & 12t + 2 \\ 6t + 5 & 6t + 7 & & 12t + 3 \end{array}$$

to color the vertices of each K_4 , and color the vertices of H_1 clockwise as $6t + 1$, $12t + 4$, $6t + 2$, $12t + 5$, $6t + 3$, $12t + 6$. Then H_1 has no bad edges and each K_4 has two. If H_2 is chosen, use the columns of

$$\begin{array}{cc|cc|cc} 1 & 2t - 1 & 2t + 2 & 4t & 4t + 3 & 6t + 1 \\ 2 & \dots & 2t & 2t + 3 & \dots & 4t + 1 & 4t + 4 & \dots & 6t + 2 \\ 6t + 4 & 8t + 2 & 8t + 5 & 10t + 3 & 10t + 6 & 12t + 4 \\ 6t + 5 & 8t + 3 & 8t + 6 & 10t + 4 & 10t + 7 & 12t + 5 \end{array}$$

to color the K_4 's, and color the vertices of H_2 clockwise from its top left vertex as $2t + 1$, $8t + 4$, $4t + 2$, $12t + 6$, $6t + 3$, $10t + 5$. Then H_2 has no bad edges and each K_4 has two.

We next verify $M^* = k$ for **C** when $k = 3t$ and $\alpha = 4t$. If H_2 is chosen, color the K_4 's by the columns of

$$\begin{array}{cccccc} 3 & 4 & 5 & & 4t - 1 & 4t & 4t + 1 \\ 4t + 4 & 4t + 5 & 6 & \dots & 8t & 8t + 1 & 8t + 2 \\ 8t + 5 & 4t + 6 & 4t + 7 & & 12t + 1 & 12t + 3 & 12t + 5 \\ 8t + 6 & 8t + 7 & 8t + 8 & & 12t + 2 & 12t + 4 & 12t + 6 \end{array}$$

and color H_2 's vertices clockwise from its top left vertex as 1, $4t + 3$, $8t + 4$, 2, $8t + 3$, $4t + 2$. Then every K_4 has one bad edge and H_2 has none. If H_1 is chosen, color the K_4 's by the columns of

$$\begin{array}{cccccc} 2 & 4 & 6 & 7 & & 4t + 1 & 4t + 2 \\ 4t + 3 & 4t + 5 & 4t + 7 & 8 & \dots & 8t + 2 & 8t + 3 \\ 4t + 4 & 8t + 7 & 4t + 8 & 4t + 9 & & 12t + 3 & 12t + 5 \\ 8t + 6 & 8t + 8 & 8t + 9 & 8t + 10 & & 12t + 4 & 12t + 6 \end{array}$$

and color H_1 's vertices clockwise as 1, $4t + 6$, 3, $8t + 4$, 5, $8t + 5$. Then every K_4 has one bad edge and H_1 has none.

The remaining cases for **C** with maximum $\alpha = \lceil (4k + 1)/3 \rceil$ are

$$\begin{array}{l} k = 3t + 1, \quad t \geq 1, \quad n = 12t + 10, \quad \alpha = 4t + 2; \\ k = 3t + 2, \quad t \geq 0, \quad n = 12t + 14, \quad \alpha = 4t + 3. \end{array}$$

If H_1 is chosen with $k = 3t + 1$, color the K_4 's by the columns of

$$\begin{array}{cccccc} 2 & 4 & 6 & 7 & & 4t+2 & 4t+3 \\ 4t+5 & 4t+7 & 4t+9 & 8 & \cdots & 8t+5 & 4t+4 \\ 4t+6 & 8t+10 & 8t+12 & 4t+11 & & 8t+6 & 8t+7 \\ 8t+9 & 8t+11 & 8t+13 & 8t+14 & & 12t+9 & 12t+10 \end{array}$$

and color H_1 's vertices clockwise as 1, $4t + 8$, 3, $4t + 10$, 5, $8t + 8$. If H_2 is chosen with $k = 3t + 2$, color the K_4 's by the columns of

$$\begin{array}{cccccc} 2 & 4 & 6 & 7 & & 4t+3 & 4t+5 \\ 4t+6 & 4t+8 & 4t+10 & 8 & \cdots & 4t+4 & 8t+9 \\ 4t+7 & 8t+13 & 4t+11 & 4t+12 & & 8t+8 & 12t+13 \\ 8t+12 & 8t+14 & 8t+15 & 8t+16 & & 12t+12 & 12t+14 \end{array}$$

and color H_1 's vertices clockwise as 1, $4t + 9$, 3, $8t + 10$, 5, $8t + 11$. In both cases every K_4 has one bad edge and H_1 has none.

If H_2 is chosen with $k = 3t + 1$, color the K_4 's by the columns of

$$\begin{array}{cccccc} 3 & 4 & 5 & & & 4t+3 \\ 4t+6 & 4t+7 & 6 & \cdots & & 8t+6 \\ 8t+9 & 4t+8 & 4t+9 & & & 12t+9 \\ 8t+10 & 8t+11 & 8t+12 & & & 12t+10 \end{array}$$

and color H_2 's vertices clockwise from its top left vertex as 1, $4t + 5$, 2, $8t + 8$, $4t + 4$, $8t + 7$. If H_2 is chosen with $k = 3t + 2$, color the K_4 's by the columns of

$$\begin{array}{cccccc} 3 & 4 & 5 & & & 4t+4 \\ 4t+7 & 4t+8 & 6 & \cdots & & 8t+8 \\ 8t+11 & 4t+9 & 4t+10 & & & 12t+13 \\ 8t+12 & 8t+13 & 8t+14 & & & 12t+14 \end{array}$$

and color H_2 's vertices clockwise from its top left vertex as 1, $4t + 6$, 2, $8t + 10$, $4t + 5$, $8t + 9$. In both cases every K_4 has one bad edge and H_2 has none. ■

REFERENCES

- [1] P. C. FISHBURN, J. H. KIM, J. C. LAGARIAS, AND P. E. WRIGHT, *Interference-minimizing colorings of regular graphs*, SIAM J. Discrete Math., 11 (1998), pp. 15–40.
- [2] P. FISHBURN AND P. E. WRIGHT, *Interference patterns in bijective colorings of 2-regular graphs*, Discrete Appl. Math., 102 (2000), pp. 189–204.
- [3] J. R. GRIGGS AND D. D.-F. LIU, *The channel assignment problem for mutually adjacent sites*, J. Combin. Theory Ser. A, 68 (1994), pp. 169–183.
- [4] D. R. GUICHARD AND J. W. KRUSSEL, *Pair labelings of graphs*, SIAM J. Discrete Math., 5 (1992), pp. 144–149.
- [5] W. K. HALE, *Frequency assignment: Theory and application*, Proc. IEEE, 68 (1980), pp. 1497–1514.
- [6] A. RAYCHAUDHURI, *Further results on T-coloring and frequency assignment problems*, SIAM J. Discrete Math., 7 (1994), pp. 605–613.
- [7] F. S. ROBERTS, *T-colorings of graphs: Recent results and open problems*, Discrete Math., 93 (1991), pp. 229–245.
- [8] B. A. TESMAN, *List T-colorings of graphs*, Discrete Appl. Math., 45 (1993), pp. 277–289.
- [9] A. VINCE, *Star chromatic number*, J. Graph Theory, 12 (1988), pp. 551–559.

STRONG T-PERFECTION OF BAD- K_4 -FREE GRAPHS*

ALEXANDER SCHRIJVER[†]

Abstract. We show that each graph not containing a bad subdivision of K_4 as a subgraph is strongly t-perfect. Here a graph $G = (V, E)$ is *strongly t-perfect* if, for each weight function $w : V \rightarrow \mathbb{Z}_+$, the maximum weight of a stable set is equal to the minimum (total) cost of a family of vertices, edges, and circuits covering any vertex v at least $w(v)$ times. By definition, the *cost* of a vertex or edge is 1, and the *cost* of a circuit C is $\lfloor \frac{1}{2}|VC| \rfloor$. A subdivision of K_4 is called *bad* if each triangle has become an odd circuit and if it is not obtained by making the edges in a 4-circuit of K_4 evenly subdivided, while the other two edges are not subdivided.

The theorem generalizes earlier results of Gerards [*J. Combin. Theory Ser. B*, 47 (1989), pp. 330–348] on the strong t-perfection of odd- K_4 -free graphs and of Gerards and Shepherd [*SIAM J. Discrete Math.*, 11 (1998), pp. 524–545] on the t-perfection of bad- K_4 -free graphs.

Key words. t-perfect, graph, stable set, polytope

AMS subject classifications. 05C69, 90C27, 90C57

PII. S0895480101401101

1. Introduction. A graph $G = (V, E)$ is called *t-perfect* if the stable set polytope of G (= the convex hull of the incidence vectors in \mathbb{R}^V of stable sets) is determined by

$$(1.1) \quad \begin{array}{ll} \text{(i)} & 0 \leq x_v \leq 1 & \text{for each } v \in V; \\ \text{(ii)} & x_u + x_v \leq 1 & \text{for each edge } uv \in E; \\ \text{(iii)} & x(VC) \leq \lfloor \frac{1}{2}|VC| \rfloor & \text{for each odd circuit } C. \end{array}$$

Here $x(U) := \sum_{v \in U} x_v$ for any $U \subseteq V, E$, and V, E denote the sets of vertices and edges, respectively, of G . A circuit C is *odd* (*even*) if $|VC|$ is odd (even).

A motivation for the concept of t-perfection lies in the fact that a linear function $w^T x$ can be maximized over (1.1) in strongly polynomial time (with the ellipsoid method, since the separation problem over (1.1) is polynomial-time solvable). Hence a maximum-weight stable set in a t-perfect graph can be found in strongly polynomial time.

G is called *strongly t-perfect* if system (1.1) is totally dual integral—that is, if for each weight function $w : V \rightarrow \mathbb{Z}_+$, the linear program of maximizing $w^T x$ over (1.1) has an integer optimum dual solution. This implies that it also has an integer optimum primal solution. In particular, all vertices of the polytope determined by (1.1) are integer, and hence the polytope is the stable set polytope. So strong t-perfection implies t-perfection.

Strong t-perfection can be characterized equivalently as follows. For any $w : V \rightarrow \mathbb{Z}_+$, let $\alpha_w(G)$ denote the maximum weight of a stable set in G . Define a *w-cover* as a family of vertices, edges, and odd circuits such that each vertex v is covered at least $w(v)$ times. (In a *family*, repetition is allowed.) By definition, the *cost* of a vertex or edge is 1, the *cost* of a circuit C is $\lfloor \frac{1}{2}|VC| \rfloor$, and the *cost* of a *w-cover* is the sum

*Received by the editors July 24, 2001; accepted for publication (in revised form) September 21, 2001; published electronically June 5, 2002.

<http://www.siam.org/journals/sidma/15-3/40110.html>

[†]CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, and Department of Mathematics, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands (lex@cwi.nl).

of the costs of its elements (counting multiplicities). Let $\tilde{\rho}_w(G)$ denote the minimum cost of a w -cover. Then

(1.2) a graph G is strongly t -perfect if and only if $\alpha_w(G) = \tilde{\rho}_w(G)$ for each $w : V \rightarrow \mathbb{Z}_+$.

The classes of t -perfect and strongly t -perfect graphs are closed under taking induced subgraphs. However, no characterization is known in terms of forbidden induced subgraphs.

If we also take noninduced subgraphs, the situation is clearer (although it does not yield a characterization). Here subdivisions of K_4 come in. A K_4 -subdivision H is called *odd*, or just *an odd K_4* , if each triangle of K_4 has become an odd circuit in H . It was shown by Gerards [6] that

(1.3) each graph without odd K_4 is strongly t -perfect.

(By “a graph without” odd K_4 we mean a graph not containing an odd K_4 as subgraph.) It extends an earlier result of Gerards and Schrijver [7] that such graphs are t -perfect.

There exist, however, odd K_4 's that are t -perfect. Following Gerards and Shepherd [8], we call an odd K_4 -subdivision a *bad K_4* if it does *not* have the following property:

(1.4) the edges of K_4 that have become an even path form a 4-cycle in K_4 ,
while the two other edges of K_4 are not subdivided.

This name is motivated by the fact, shown by Barahona and Mahjoub [1], that a subdivision of K_4 is t -perfect if and only if it is not a bad K_4 . Gerards and Shepherd [8] proved that

(1.5) each graph without bad K_4 is t -perfect.

(Gerards and Shepherd [8] also showed that graphs without bad K_4 can be recognized in polynomial time.)

In the present paper, we show more strongly that these graphs are strongly t -perfect. This generalizes (1.3) and (1.5), and implies for any graph G that

(1.6) each subgraph of G is t -perfect
 \iff each subgraph of G is strongly t -perfect
 $\iff G$ has no bad K_4 as subgraph.

On the other hand, there exist strongly t -perfect graphs that contain a bad K_4 ; see Figure 1.1.

Our proof method was inspired by a method of Geelen and Guenin [5] for proving a special case of a theorem of Seymour [12] on packing the edge sets of odd circuits in odd- K_4 -free graphs.

The above results contain the strong t -perfection of series-parallel graphs, which are, as is well known, those graphs not containing any K_4 -subdivision (Boulala and Uhry [2]), and of almost bipartite graphs—graphs G having a vertex v with $G - v$ bipartite (Fonlupt and Uhry [4], Sbihi and Uhry [10]).

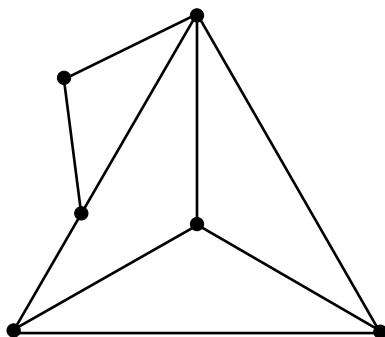


FIG. 1.1.

A related theorem was proved by Sewell and Trotter [11]. A K_4 -subdivision is called a *totally odd K_4* if it arises from K_4 by replacing each edge by an odd path. The theorem says that a graph G without totally odd K_4 satisfies $\alpha_{\mathbf{1}}(G) = \tilde{\rho}_{\mathbf{1}}(G)$, where $\mathbf{1}$ denotes the all-one weight function. This result does not follow from our methods.

The totally odd K_4 's are precisely those K_4 -subdivisions G with $\alpha_{\mathbf{1}}(G) < \tilde{\rho}_{\mathbf{1}}(G)$. So the theorem of Sewell and Trotter and the theorem presented in this paper suggest the question of whether, for each graph G and each $w : VG \rightarrow \mathbb{Z}_+$ with $\alpha_w(G) < \tilde{\rho}_w(G)$, G contains a K_4 -subdivision H as subgraph such that $\alpha_{w'}(H) < \tilde{\rho}_{w'}(H)$, where $w' := w|_{VH}$. The answer is unknown.

To complete the picture, it was shown by Zang [15] and Thomassen [13] that $\chi(G) \leq 3$ for any graph G without totally odd K_4 . This was conjectured by Toft [14], and was proved by Hadwiger [9] for series-parallel graphs, by Catlin [3] for odd- K_4 -free graphs, and by Gerards and Shepherd [8] for bad- K_4 -free graphs. (However, there exist strongly t-perfect graphs G with $\chi(G) > 3$.)

A.M.H. Gerards and P.D. Seymour proved in 1991 (personal communication) that if G contains no odd K_4 , then the stable set polytope of G has the integer decomposition property. In other words, any $w : VG \rightarrow \mathbb{Z}_+$ is the sum of the incidence vectors of k stable sets, where k is the minimum integer for which $\frac{1}{k}w$ belongs to the stable set polytope. It implies the result of Catlin mentioned above.

2. Graphs without bad K_4 . In this section we prove a technical lemma on bad- K_4 -free graphs. Let G be graph without bad K_4 , and let C be an even circuit in G . Let e_1, \dots, e_n be chords of C such that e_i has ends s_i and s_{n+i} (say) (for $i = 1, \dots, n$), such that s_1, \dots, s_{2n} are distinct and occur in this order clockwise along C , and such that, for each $i = 1, \dots, 2n$, the clockwise $s_{i-1} - s_i$ path R_i along C has even length. (We take indices mod $2n$ and set $e_{n+i} := e_i$ for $i = 1, \dots, n$.) Define $D := \{e_1, \dots, e_n\}$.

Call a path B in G a *bow* if B is simple, has length at least 2, and intersects C precisely in its end vertices. We call a bow an *odd bow* if it forms with a subpath of C an odd circuit and an *even bow* if it forms with a subpath of C an even circuit. (So an odd (even) bow need not be an odd (even) path. To avoid confusion, we therefore do not use the more familiar term “ear.”)

We will study in particular the occurrence of odd bows. We say that a bow B *crosses* an edge $e \in D$ (and conversely) if e is disjoint from the ends a, b (say) of B

and connects distinct components of the graph $C - a - b$. Then

$$(2.1) \quad \text{an odd bow } B \text{ does not cross any edge } e \text{ in } D.$$

Otherwise, C , B , and e form a bad K_4 , a contradiction.

Equation (2.1) implies that the ends of any odd bow belong to VR_j for some $j = 1, \dots, 2n$. Define

$$(2.2) \quad J := \{j \in \{1, \dots, 2n\} \mid \text{there exists an odd bow with ends in } VR_j\}.$$

We prove the following lemma.

LEMMA 2.1. *There exists an $i \in \{1, \dots, 2n\}$ such that $i+1, i+2, \dots, i+n-1 \notin J$.*

Proof. Consider a counterexample with n as small as possible. Define $L := \{i \mid i+2, \dots, i+n-1 \notin J\}$. Then, for each i ,

$$(2.3) \quad i \in L \text{ or } i+n \in L.$$

To see this, by symmetry it suffices to show this for $i = n$. Delete e_n . By the minimality of n , the lemma holds for the new structure. In the new structure, the paths R_n and R_{n+1} have merged to one path, and similarly the path R_{2n} and R_1 have merged to one path. If (2.3) does not hold for the original structure, then, for some $i \in \{2, \dots, n-1\}$, there is no odd bow with ends in one of $VR_{i+1}, \dots, VR_{n-1}, VR_n \cup VR_{n+1}, VR_{n+2}, \dots, VR_{i+n-1}$ or there is no odd bow with ends in one of $VR_{i+n+1}, \dots, VR_{2n-1}, VR_{2n} \cup VR_1, VR_2, \dots, VR_{i-1}$. Either case implies the lemma for the original structure, a contradiction. So we have (2.3).

We derive from this that $n = 2$. As the lemma does not hold, we know that $i \notin L$ or $i+1 \notin L$ for each i . Hence, by (2.3), $i \in L$ or $i+1 \in L$ for each i . So the indices i are alternatingly in and out of L . If $n \geq 4$, then we can assume that each even i belongs to L , and hence, by the definition of L , $J = \emptyset$, a contradiction.

So $n \leq 3$. Suppose $n = 3$. We may assume $J = \{1, 3, 5\}$. For $j = 1, 3, 5$, let B_j be an odd bow with ends in VR_j . Then B_1, B_3, B_5 are pairwise disjoint, for suppose that (say) B_1 and B_3 have a vertex in common. Choose an end a of B_1 with $a \neq s_1$. Follow B_1 from a until we reach B_3 . We can continue along B_3 so as to create an odd bow B (as B_3 is an odd bow). As B crosses e_1 , this contradicts (2.1).

So B_1, B_3, B_5 are pairwise disjoint. Let R'_j be obtained from R_j by replacing part of R_j by B_j . Then $R'_1, R_2, R'_3, R_4, R'_5$ and e_1, e_2, e_3 form a bad K_4 , a contradiction.

So $n = 2$. As the lemma does not hold, we know $J = \{1, 2, 3, 4\}$. For $j = 1, \dots, 4$, let B_j be an odd bow with ends in VR_j . If the B_j are pairwise internally vertex-disjoint, we obtain a bad K_4 , a contradiction. So at least two of the B_j have an internal vertex in common. Define $S := \{s_1, s_2, s_3, s_4\}$. To analyze this, we first prove the following:

$$(2.4) \quad \text{Let } B \text{ be a bow with ends } a, b \text{ and } a \in VR_1 \setminus S \text{ and } b \notin VR_1.$$

Then a and b are equal to the middle vertices of R_1 and R_3 , respectively.

By (2.1), B is an even bow. By symmetry, we can assume that $b \in VR_2 \cup VR_3 \setminus \{s_1, s_3\}$. Let C' be the (even) circuit obtained from C by replacing the $a - b$ path P along C that traverses s_1 , by B . Let e'_1 be the extension of e_1 with the $s_1 - a$ part of R_1 . So e'_1 is an odd bow of C' . If $b \in VR_2$, then e_2 is a chord of C' that crosses e'_1 , contradicting (2.1). So $b \in VR_3 \setminus S$.

Let e'_2 be the extension of e_2 with the $s_2 - b$ part of R_3 . Again, e'_2 is an odd bow of C' . Then C', e'_1, e'_2 form an odd K_4 -subdivision H , with trivalent vertices a, b, s_3 ,

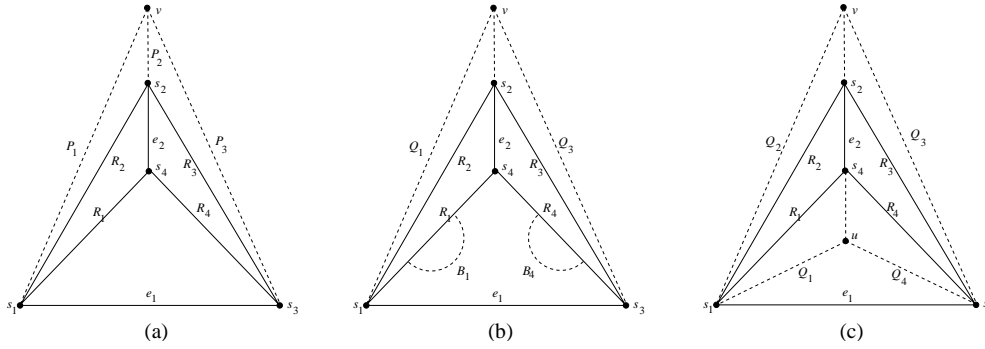


FIG. 2.1.

and s_4 . As H is not bad, and as s_4 is nonadjacent (in H) to b and s_3 , we know that s_4 is adjacent (in H) to a . By symmetry, a is adjacent to s_1 , and b to s_2 and to s_3 . This gives (2.4).

From this we derive the following:

Let T be a tree with three end vertices a, b, c , and trivalent vertex v such that T has only its end vertices in common with C and such that a, b, c do not all belong to some VR_i ($i = 1, \dots, 4$). Then for some i , $\{a, b, c\} = \{s_{i-1}, s_i, s_{i+1}\}$, s_i is adjacent (2.5) to v , and the $v - s_{i-1}$ and $v - s_{i+1}$ paths along T are even.

We first show that $a, b, c \in S$. Suppose not. Then we can assume $a \in VR_1 \setminus S$. Since a, b, c not all belong to VR_1 , we can assume that $b \notin VR_1$. Then by (2.4), a and b are the middle vertices of R_1 and R_3 , respectively. By symmetry of a and b , we can assume that $c \notin VR_1$, implying similarly that $c = b$, a contradiction. So $a, b, c \in S$.

Next we can assume that $\{a, b, c\} = \{s_1, s_2, s_3\}$. Let P_i be the $v - s_i$ path in T (for $i = 1, 2, 3$) (cf. Figure 2.1(a)). As P_1 and P_3 form a bow connecting s_1 and s_3 , it is an even bow and we have $|EP_1| \equiv |EP_3| \pmod{2}$. If, moreover, $|EP_1| \equiv |EP_2| \pmod{2}$, then $P_1, P_2, P_3, R_1, R_4, e_1$, and e_2 form a bad K_4 . So $|EP_1| \not\equiv |EP_2| \pmod{2}$. Then P_1, P_2, P_3, R_2, R_3 , and e_1 form an odd K_4 . As it is not bad and as e_1 has length 1, we have $|EP_2| = 1$, implying (2.5).

This implies that

$$(2.6) \quad G - VC \text{ has no component } K \text{ with } s_1, s_2, s_3, s_4 \in N(K).$$

Otherwise, there is a tree T intersecting VC only in its end vertices s_1, s_2, s_3, s_4 . By (2.5), the neighbor v_i of any s_i in T has degree at least 3 (by considering a subtree with ends s_{i-1}, s_i, s_{i+1}). It also follows from (2.5) that $v_i \neq v_{i+1}$ for each i . So $v_1 = v_3$, contradicting (2.5) (by considering a subtree with ends s_1, s_2, s_3). This gives (2.6).

This implies that B_1 and B_3 are disjoint. Otherwise, by (2.5), the ends of B_1 and B_3 are s_1, s_2, s_3, s_4 , contradicting (2.6). Similarly, B_2 and B_4 are disjoint.

So we can assume that B_2 and B_3 have a vertex in common, and hence, by (2.5), that there is a vertex $v \notin VC$ adjacent to s_2 and a $v - s_1$ path Q_2 and a $v - s_3$ path Q_3 such that, for $i = 2, 3$, B_i is the concatenation of the edge s_2v and Q_i (cf. Figure 2.1(b)).

By (2.6), neither B_1 nor B_4 has an internal vertex in common with B_2 and B_3 . If B_1 and B_4 are internally vertex-disjoint, then $B_1, B_4, e_1, e_2, vs_2, Q_1, Q_2$, and parts of R_1 and R_4 form a bad K_4 .

So B_1 and B_4 are not internally vertex-disjoint. Hence, by (2.5), there is a vertex $u \notin VC$ adjacent to s_4 and a $u - s_1$ path Q_1 and a $u - s_3$ path Q_4 such that, for $i = 1, 4$, B_i is the concatenation of the edge s_4u and Q_i (cf. Figure 2.1(c)). Then $Q_1, \dots, Q_4, vs_2, us_4, e_2$, and e_1 form a bad K_4 , a contradiction. \square

3. Strong t-perfection of bad- K_4 -free graphs. We now prove our main theorem.

THEOREM 3.1. *A graph without bad K_4 is strongly t-perfect.*

Proof. Let $G = (V, E)$ be a counterexample with $|V| + |E|$ minimum. For any weight function $w : V \rightarrow \mathbb{Z}_+$, denote $\alpha_w := \alpha_w(G)$ and $\tilde{\rho}_w := \tilde{\rho}_w(G)$. For any subset U of V let χ^U be the incidence vector of U . So for an edge $e = uv$, χ^e is the $0, 1$ vector in \mathbb{R}^V having 1's in positions u and v .

We first show the following claim.

Claim 1. There is a $w : V \rightarrow \mathbb{Z}_+$ and an edge f such that

$$(3.1) \quad \tilde{\rho}_{w+\chi^f} = \alpha_w + 1 = \tilde{\rho}_w$$

and such that

$$(3.2) \quad \alpha_{w-\chi^{VC}} = \tilde{\rho}_{w-\chi^{VC}}$$

for each odd circuit C .

Proof of Claim 1. Choose a vertex u . For any $w : V \rightarrow \mathbb{Z}_+$ with $\alpha_w < \tilde{\rho}_w$ one has

$$(3.3) \quad w(u) < w(N(u))$$

(where $N(u)$ denotes the set of neighbors of u). Otherwise, by the minimality of G , setting $G' := G - u - N(u)$ and $w' := w|VG'$,

$$(3.4) \quad \alpha_w(G) = w(u) + \alpha_{w'}(G') = w(u) + \tilde{\rho}_{w'}(G') \geq \tilde{\rho}_w(G),$$

since $G[\{u\} \cup N(u)]$ has a $w|N(u) \cup \{u\}$ -cover of cost $w(u)$ (as $w(u) \geq w(N(u))$). Equation (3.4) contradicts our assumption, which proves (3.3).

By (3.3), we can choose w such that $\alpha_w < \tilde{\rho}_w$ and such that $w(V \setminus \{u\}) - w(u)$ is as small as possible. Then

$$(3.5) \text{ there exists a } z \in \mathbb{Z}_+^{\delta(u)} \text{ such that for } \tilde{w} := w + \sum_{e \in \delta(u)} z_e \chi^e \text{ we have } \alpha_{\tilde{w}} = \tilde{\rho}_{\tilde{w}}.$$

To see this, it suffices to show that

$$(3.6) \quad \begin{aligned} &\text{there exists a } z \in \mathbb{Z}^{\delta(u)} \text{ and a stable set } S \text{ such that } \tilde{w} := w + \sum_{e \in \delta(u)} z_e \chi^e \text{ is} \\ &\text{nonnegative and such that } \tilde{w}(S) = \tilde{\rho}_{\tilde{w}} \text{ and } S \text{ intersects each edge incident with } u. \end{aligned}$$

This suffices, since if z' arises from z by replacing the negative entries by 0, and

$$(3.7) \quad w' := w + \sum_{e \in \delta(u)} z'_e \chi^e,$$

then $w'(S) = \tilde{w}(S) - \sum(z_e | z_e < 0)$ and $\tilde{\rho}_{w'} \leq \tilde{\rho}_{\tilde{w}} - \sum(z_e | z_e < 0)$, as $w' = \tilde{w} - \sum(z_e \chi^e | z_e < 0)$. This implies (3.5).

To prove (3.6), first suppose that $N(u)$ is a stable set. Let G' be the graph obtained from G by contracting the edges in $\delta(u)$. Then G' contains no bad K_4 . Let t be the new vertex. Let $w' : VG' \rightarrow \mathbb{Z}_+$ be defined by $w'(t) := w(N(u)) - w(u)$ and $w'(v) := w(v)$ if $v \neq t$. Since G' is smaller than G , we know $\alpha_{w'}(G') = \tilde{\rho}_{w'}(G')$.

Consider a w' -cover \mathcal{F}' in G' of cost $\tilde{\rho}_{w'}(G')$. Let λ be the number of circuits in \mathcal{F}' that are not circuits in G . So they traverse t and can be made to circuits in G by adding two edges incident with u . It gives, for some \tilde{w} , a \tilde{w} -cover \mathcal{F} in G of cost $\tilde{\rho}_{w'}(G') + \lambda$ such that \tilde{w} coincides with w on $V \setminus (N(u) \cup \{u\})$ and such that $\tilde{w}(u) = \lambda$ and $\tilde{w}(N(u)) = w'(t) + \lambda$. Hence the cost is $\tilde{\rho}_{w'}(G') + \tilde{w}(u)$ and $\tilde{w}(N(u)) - \tilde{w}(u) = w(N(u)) - w(u)$. This last implies that $\tilde{w} = w + \sum_{e \in \delta(u)} z_e \chi^e$ for some $z \in \mathbb{Z}^{\delta(u)}$.

Now let S' be a stable set in G' with $w'(S') = \alpha_{w'}(G')$. If $t \in S'$, define $S := (S' \setminus \{t\}) \cup N(u)$, and if $t \notin S'$, define $S := S' \cup \{u\}$. So S is a stable set in G . Then $w(S) = w'(S') + w(u)$ and S intersects each edge incident with u . So

$$(3.8) \quad \tilde{w}(S) = w'(S') + \tilde{w}(u) = \tilde{\rho}_{w'}(G') + \tilde{w}(u) \geq \tilde{\rho}_{\tilde{w}}(G).$$

This gives (3.6) in case $N(u)$ is a stable set.

If $N(u)$ is not a stable set, let $G' := G - u - N(u)$ and $w' := w|VG'$. By the minimality of G , $\alpha_{w'}(G') = \tilde{\rho}_{w'}(G')$. Let \mathcal{F}' be a w' -cover in G' of cost $\tilde{\rho}_{w'}(G')$. By adding to \mathcal{F}' a number of times a triangle incident with u we obtain a \tilde{w} -cover \mathcal{F} in G for some $\tilde{w} : V \rightarrow \mathbb{Z}_+$, where \tilde{w} coincides with w on $V \setminus (\{u\} \cup N(u))$, where $\tilde{w}(N(u)) - \tilde{w}(u) = w(N(u)) - w(u)$, and where \mathcal{F} has cost $\tilde{\rho}_{w'}(G') + \tilde{w}(u)$.

Now let S' be a stable set in G' with $w'(S') = \alpha_{w'}(G')$. Define $S := S' \cup \{u\}$. So S is a stable set in G . Then $w(S) = w'(S') + w(u)$ and S intersects each edge incident with u . Moreover, $\tilde{w}(S) = w'(S') + \tilde{w}(u) = \tilde{\rho}_{w'}(G') + \tilde{w}(u) \geq \tilde{\rho}_{\tilde{w}}(G)$. So we have (3.6), and hence (3.5).

Choose z in (3.5) with $z(\delta(u))$ as small as possible. Choose $f \in \delta(u)$ with $z_f \geq 1$. We can assume that $z_f = 1$ and $z_e = 0$ for all other edges e , as we can reset $w := \tilde{w} - \chi^f$. (This resetting does not change the value of $w(V \setminus \{u\}) - w(u)$.) Then (3.2) follows from the minimality of $w(V \setminus \{u\}) - w(u)$.

We finally show (3.1). By the definition of z , $\tilde{\rho}_{w+\chi^f} = \alpha_{w+\chi^f}$. Also we have $\alpha_{w+\chi^f} \leq \alpha_w + 1$, since any stable set S satisfies $(w + \chi^f)(S) \leq w(S) + 1$. As $\tilde{\rho}_w \leq \tilde{\rho}_{w+\chi^f}$, this implies (3.1). *End of Proof of Claim 1.*

As of now we assume that w and f satisfy (3.1) and (3.2). Let f connect vertices u and u' . Since by the minimality of G , G has no isolated vertices, there exists a minimum-cost $w + \chi^f$ -cover consisting only of edges and odd circuits, say, $e_1, \dots, e_t, C_1, \dots, C_k$. We choose f and $e_1, \dots, e_t, C_1, \dots, C_k$ such that

$$(3.9) \quad |VC_1| + \dots + |VC_k|$$

is as small as possible. Then

$$(3.10) \quad \text{at least two of the } C_i \text{ traverse } f.$$

To see this, let $G' := G - f$. If $\alpha_w(G') = \alpha_w(G)$, then by induction G' has a w -cover of cost α_w . As this is a w -cover in G as well, this would imply $\alpha_w = \tilde{\rho}_w$, a contradiction.

So $\alpha_w(G') > \alpha_w(G)$. That is, there exists a stable set S in G' with $w(S) > \alpha_w$. Necessarily, S contains both u and u' . Then, for any circuit C traversing f ,

$$(3.11) \quad |VC \cap S| \leq \lfloor \frac{1}{2}|VC| \rfloor + 1.$$

Also, f is not among e_1, \dots, e_t , since otherwise $\tilde{\rho}_w \leq \tilde{\rho}_{w+\chi^f} - 1$, contradicting (3.1). Setting l to the number of C_i traversing f , we obtain

$$(3.12) \quad \begin{aligned} \tilde{\rho}_{w+\chi^f} &\leq \alpha_w + 1 \leq w(S) = (w + \chi^f)(S) - 2 \leq -2 + \sum_{j=1}^t |e_j \cap S| + \sum_{i=1}^k |VC_i \cap S| \\ &\leq -2 + t + \sum_{i=1}^k \lfloor \frac{1}{2} |VC_i| \rfloor + l = \tilde{\rho}_{w+\chi^f} + l - 2. \end{aligned}$$

So $l \geq 2$, which is (3.10).

By (3.10) we can assume that C_1 and C_2 traverse f . It is convenient to assume that $EC_1 \setminus \{f\}$ and $EC_2 \setminus \{f\}$ are disjoint; this can be achieved by adding parallel edges. So $EC_1 \cap EC_2 = \{f\}$.

Then,

$$(3.13) \quad \begin{aligned} \text{if } C \text{ is an odd circuit with } EC \subseteq EC_1 \cup EC_2, \text{ then } f \in EC \text{ and } EC_1 \Delta EC_2 \Delta EC \\ \text{again is an odd circuit.} \end{aligned}$$

To see this, define $C'_1 := C$. As $EC_1 \Delta EC_2 \Delta EC$ is an odd cycle (a *cycle* is an edge-disjoint union of circuits), it can be decomposed into circuits C'_2, \dots, C'_p , with C'_2, \dots, C'_q odd and C'_{q+1}, \dots, C'_p even ($q \geq 2$). Choose for each $i = q+1, \dots, p$ a perfect matching M_i in C'_i . Let e'_1, \dots, e'_r be the edges in the matchings M_i and in $\{f\} \setminus EC$. Then

$$(3.14) \quad \chi^{VC_1} + \chi^{VC_2} = \sum_{i=1}^q \chi^{VC'_i} + \sum_{j=1}^r \chi^{e'_j}$$

and

$$(3.15) \quad \begin{aligned} \lfloor \frac{1}{2} |VC_1| \rfloor + \lfloor \frac{1}{2} |VC_2| \rfloor &= \frac{1}{2} |EC_1| + \frac{1}{2} |EC_2| - 1 = r - 1 + \frac{1}{2} \sum_{i=1}^q |EC'_i| \\ &\geq r + \sum_{i=1}^q \lfloor \frac{1}{2} |VC'_i| \rfloor. \end{aligned}$$

So replacing C_1, C_2 by C'_1, \dots, C'_q and adding e'_1, \dots, e'_r to e_1, \dots, e_t again gives a $w + \chi^f$ -cover of cost at most $\tilde{\rho}_{w+\chi^f}$.

If $f \notin EC$, then f is among e'_1, \dots, e'_r . Hence deleting f gives a w -cover of cost at most $\tilde{\rho}_{w+\chi^f} - 1 \leq \alpha_w$, contradicting (3.1). So $f \in EC$. As this is true for any odd circuit in $EC_1 \cup EC_2$ we know that $f \in EC'_i$ for $i = 1, \dots, q$ and that $q = 2$.

If $p \geq 3$ or $r \geq 1$, then $|EC'_1| + |EC'_2| < |EC_1| + |EC_2|$, contradicting the minimality of (3.9). This proves (3.13).

First, it implies

$$(3.16) \quad \text{a circuit in } EC_1 \cup EC_2 \text{ is odd if and only if it contains } f.$$

A second consequence is as follows. Let P_i be the $u - u'$ path $C_i \setminus \{f\}$. Orient the edges occurring in the path $P_i := C_i \setminus \{f\}$ in the direction from u to u' for $i = 1, 2$. Then

$$(3.17) \quad \text{the orientation is acyclic.}$$

For suppose there exists a directed circuit C . Then $(EC_1 \cup EC_2) \setminus EC$ contains a directed $u - u'$ path, and hence an odd circuit C' . Hence, by (3.13), $EC_1 \triangle EC_2 \triangle EC'$ is an odd circuit, however, containing the even circuit EC , a contradiction.

Let A and B be the color classes of the bipartite graph $(VP_1 \cup VP_2, EP_1 \cup EP_2)$ such that $u, u' \in A$. So

$$(3.18) \quad \begin{aligned} A &:= \{v \in VP_1 \cup VP_2 \mid \text{there exists an even-length directed } u - v \text{ path}\}, \\ B &:= \{v \in VP_1 \cup VP_2 \mid \text{there exists an odd-length directed } u - v \text{ path}\}. \end{aligned}$$

Define

$$(3.19) \quad \begin{aligned} X &:= VP_1 \cap VP_2 \\ \text{and } U &:= \left\{ v \in V \mid w(v) = \sum_{j=1}^t |e_j \cap \{v\}| + \sum_{j=1}^k |VC_j \cap \{v\}| \right\}. \end{aligned}$$

We next show the following technical, but straightforward to prove, claim.

Claim 2. Let $z \in A$, let Q be an even length directed $u - z$ path, and let S be a stable set in G . Then

$$(3.20) \quad (w - \chi^{VQ})(S) \geq \alpha_w - \lfloor \frac{1}{2} |VQ| \rfloor + 1$$

if and only if

$$(3.21) \quad \begin{aligned} \text{(i)} \quad & |e_j \cap S| = 1 \text{ for each } j = 1, \dots, t, \\ \text{(ii)} \quad & |VC_j \cap S| = \lfloor \frac{1}{2} |VC_j| \rfloor \text{ for } j = 3, \dots, k, \\ \text{(iii)} \quad & S \subseteq U, \\ \text{(iv)} \quad & S \text{ contains } B \setminus VQ \text{ and is disjoint from } A \setminus VQ, \\ \text{(v)} \quad & S \text{ contains } B \cap X \text{ and is disjoint from } A \cap X. \end{aligned}$$

Proof of Claim 2. We can assume that $EQ \subseteq EC_1$. Set $W := VC_1 \setminus VQ$. So $|W|$ is even. Consider the following sequence of (in)equalities:

$$(3.22) \quad \begin{aligned} (w - \chi^{VQ})(S) &= w(S) - |VQ \cap S| \leq (w + \chi^f)(S) - |VQ \cap S| \\ &\leq \sum_{j=1}^t |e_j \cap S| + \sum_{j=1}^k |VC_j \cap S| - |VQ \cap S| = \sum_{j=1}^t |e_j \cap S| + \sum_{j=2}^k |VC_j \cap S| + |W \cap S| \\ &\leq t + \sum_{j=2}^k \lfloor \frac{1}{2} |VC_j| \rfloor + |W \cap S| = \tilde{\rho}_{w+\chi^f} - \lfloor \frac{1}{2} |VC_1| \rfloor + |W \cap S| \\ &\leq \tilde{\rho}_{w+\chi^f} - \lfloor \frac{1}{2} |VC_1| \rfloor + \frac{1}{2} |W| = \alpha_w + 1 - \lfloor \frac{1}{2} |VQ| \rfloor. \end{aligned}$$

Hence (3.20) holds if and only if equality holds throughout in (3.22), which is equivalent to (3.21). *End of Proof of Claim 2.*

By (3.17), we can order the vertices in X as $v_0 = u, v_1, \dots, v_s = u'$ such that both P_1 and P_2 traverse them in this order. For $j = 0, \dots, s$, let \mathcal{P}_j be the collection of directed $u - x$ paths, where $x = v_j$ if $v_j \in A$ and x is an in-neighbor of v_j if $v_j \in B$. So $x \in A$.

Let j be the largest index for which there exists a path $Q \in \mathcal{P}_j$ with

$$(3.23) \quad \alpha_{w-\chi^{VQ}} \leq \alpha_w - \lfloor \frac{1}{2} |VQ| \rfloor.$$

Such a j exists, since (3.23) holds for the trivial directed $u - u$ path, as $\alpha_{w-\chi^u} \leq \alpha_w$. Also, $j < s$, since otherwise $VQ = VC$ for some odd circuit C , and hence with (3.2) we have

$$(3.24) \quad \tilde{\rho}_w \leq \tilde{\rho}_{w-\chi^{VC}} + \lfloor \frac{1}{2} |VC| \rfloor = \alpha_{w-\chi^{VC}} + \lfloor \frac{1}{2} |VC| \rfloor \leq \alpha_w,$$

contradicting (3.1).

Let Q_1 and Q_2 be the two paths in \mathcal{P}_{j+1} that extend Q . By the maximality of j , we know

$$(3.25) \quad \alpha_{w-\chi^{VQ_i}} \geq \alpha_w - \lfloor \frac{1}{2} |VQ_i| \rfloor + 1.$$

Hence there exist stable sets S_1 and S_2 with

$$(3.26) \quad (w - \chi^{VQ_i})(S_i) \geq \alpha_w - \lfloor \frac{1}{2} |VQ_i| \rfloor + 1$$

for $i = 1, 2$. So, for $i = 1, 2$, (3.21) holds for Q_i, S_i . By (3.21)(iv), S_1 and S_2 coincide on $VP_1 \cup VP_2$ except on $VQ_1 \cup VQ_2$. In other words,

$$(3.27) \quad (S_1 \Delta S_2) \cap (VP_1 \cup VP_2) \subseteq VQ_1 \cup VQ_2.$$

By (3.21)(v), S_1 and S_2 , moreover, coincide on X .

Let H be the subgraph of G induced by $S_1 \Delta S_2$. So H is a bipartite graph, with color classes $S_1 \setminus S_2$ and $S_2 \setminus S_1$. Define

$$(3.28) \quad Y_i := VQ_i \setminus VQ$$

for $i = 1, 2$. Then

$$(3.29) \quad H \text{ contains a path connecting } Y_1 \text{ and } Y_2.$$

For suppose not. Let K be the union of the components of H that intersect Y_1 . So K is disjoint from Y_2 . Define $S := S_1 \Delta K$. Then $S \cap Y_1 = S_2 \cap Y_1$ and $S \cap Y_2 = S_1 \cap Y_2$. This implies that Q, S satisfy (3.21). Hence (3.20) holds, contradicting (3.23). This proves (3.29).

Let C be the (even) circuit formed by the two directed $v_j - v_{j+1}$ paths. So Y_1 and Y_2 are subsets of VC . Let R be a shortest path in H that connects Y_1 and Y_2 ; say it connects $y_1 \in Y_1$ and $y_2 \in Y_2$.

Since $y_1, y_2 \in S_1 \Delta S_2$, we know by (3.21)(v) that $y_1, y_2 \notin X$. By (3.21)(iv), if $y_1 \in S_1 \setminus S_2$, then $y_1 \in A$ and if $y_1 \in S_2 \setminus S_1$, then $y_1 \in B$. Similarly, if $y_2 \in S_2 \setminus S_1$, then $y_2 \in A$ and if $y_2 \in S_1 \setminus S_2$, then $y_2 \in B$.

So if R is even, then y_1 and y_2 belong to different sets A, B , and if R is odd, then y_1 and y_2 belong to the same set among A, B . Hence R forms with part of C an odd circuit.

By (3.27) and as $(S_1 \Delta S_2) \cap X = \emptyset$, there exist a directed $u - v_j$ path N' and a directed $v_{j+1} - u'$ path N'' that are (vertex-)disjoint from $S_1 \Delta S_2$. N', N'' , and f make a $v_{j+1} - v_j$ path N . Then N, R , and C make an odd K_4 , with 3-valent vertices v_j, v_{j+1}, y_1, y_2 .

By assumption, it is not a bad K_4 ; that is, it satisfies (1.4). Suppose first that R has even length. Then by (1.4) N also has even length. Hence v_j and v_{j+1} belong to different sets A, B . Then by (1.4) and the symmetry of y_1 and y_2 , we may assume that y_1 is adjacent to v_j and that y_2 is adjacent to v_{j+1} . Hence, as $y_1, y_2 \in S_1 \cup S_2$, v_j and v_{j+1} do not belong to $S_1 \cap S_2$, and so $v_j, v_{j+1} \notin B$ (by (3.21)(v)), a contradiction.

So R has length 1. Hence N has length 1 as well, and v_j, v_{j+1}, y_1, y_2 lie in the same color class of the bipartition A, B of C . So we know that

$$(3.30) \quad v_j = u, v_{j+1} = u', y_1, y_2 \in A, \text{ and } R \text{ has length 1.}$$

Let D be the set of edges of G connecting two vertices in A . So $f \in D$ and $y_1 y_2 \in D$. Hence $|D| \geq 2$. We consider the edges in D as chords of the circuit C with $EC = EP_1 \cup EP_2$.

Now any edge d in D can play the same role as f , since, if C'_1 and C'_2 denote the two odd circuits in $EC \cup \{d\}$, then

$$(3.31) \quad C'_1, C'_2, C_3, \dots, C_k, e_1, \dots, e_t \text{ form a } w + \chi^d\text{-cover of cost } \tilde{\rho}_{w+\chi^d} = \tilde{\rho}_{w+\chi^f}.$$

Indeed, as $\chi^{C'_1} + \chi^{C'_2} = \chi^d + \chi^{C_1} + \chi^{C_2} - \chi^f$, the collection $C'_1, C'_2, C_3, \dots, C_k, e_1, \dots, e_t$ is a $w + \chi^d$ -cover of cost $\tilde{\rho}_{w+\chi^d}$ with $|VC'_1| + |VC'_2| + |VC_3| + \dots + |VC_k|$ at most (3.9). Hence (3.31) follows from the choice of f .

So each $d \in D$ has all the properties derived for f so far, and it would lead to the same circuit C and to the same bipartition A, B of C .

This is used to prove that

$$(3.32) \quad \text{any edge in } D \text{ crosses any chord of } C.$$

Indeed, we need only to prove this for f . However, by the minimality of (3.9) all circuits among C_1, \dots, C_k are chordless, so each chord of C crosses f .

Let $n := |D|$, and let s_1, s_2, \dots, s_{2n} be the ends of the edges in D , in cyclic order. Let f_1, \dots, f_{2n} be the edges in D incident with s_1, \dots, s_{2n} , respectively. So $f_{n+j} = f_j$ for all j (taking indices mod $2n$). For $j = 1, \dots, 2n$, let R_j be the $s_{j-1} - s_j$ path along C that does not contain any other of the vertices s_i .

By Lemma 2.1, we can assume that $2, \dots, n \notin J$, where J is as defined in (2.2). Let Q_1 be the path of the form $Q = R_{j+1}R_{j+2} \cdots R_n$ with $0 \leq j \leq n$ such that

$$(3.33) \quad \alpha_{w-\chi^{VQ}} \geq \alpha_w - \lfloor \frac{1}{2}|VQ| \rfloor + 1$$

and such that j is maximal. This path exists, since for $j = 0$ we have (3.33), as otherwise (3.24) would again yield a contradiction.

Trivially, $j < n$, since the empty path does not satisfy (3.33). Let $Q_2 := R_{j+2}R_{j+3} \cdots R_{j+1+n}$. Since Q_2 also satisfies (3.33) (as, again, (3.24) would yield a contradiction otherwise), there exist stable sets S_1 and S_2 with

$$(3.34) \quad (w - \chi^{VQ_i})(S_i) \geq \alpha_w - \lfloor \frac{1}{2}|VQ_i| \rfloor + 1$$

for $i = 1, 2$. So, for $i = 1, 2$, (3.21) holds for Q_i, S_i where we can take for f any edge not incident with an internal vertex of Q_i . By (3.21)(iv),

$$(3.35) \quad (S_1 \Delta S_2) \cap VC \subseteq VQ_1 \cup VQ_2.$$

We (re)define H as the subgraph of G induced by $S_1 \Delta S_2$. Define

$$(3.36) \quad Y_1 := VR_{j+1} \text{ and } Y_2 := VR_{n+1} \cup VR_{n+2} \cup \dots \cup VR_{n+j+1}.$$

Then

$$(3.37) \quad H \text{ contains a path connecting } Y_1 \text{ and } Y_2.$$

For suppose not. Let K be the union of the components of H that intersect Y_1 . So K is disjoint from Y_2 . Define $S := S_1 \triangle K$. Then $S \cap Y_1 = S_2 \cap Y_1$ and $S \cap Y_2 = S_1 \cap Y_2$. This implies that $Q := R_{j+2}R_{j+3} \cdots R_n$ and S satisfy (3.21), taking $f := f_n$. Hence (3.20) holds for Q , contradicting the maximality of j . This proves (3.37).

Let R be a shortest path in H that connects Y_1 and Y_2 ; say it connects $y_1 \in Y_1$ and $y_2 \in Y_2$. By (3.35), any internal vertex of R that is on C is an internal vertex of $R_{j+2}R_{j+3} \cdots R_n$. If $y_1 \in S_1 \setminus S_2$, as y_1 is not an internal vertex of Q_2 , we know $y_1 \in A$. Similarly, if $y_1 \in S_2 \setminus S_1$, then $y_1 \in B$. Similarly, if $y_2 \in S_2 \setminus S_1$, then $y_2 \in A$, and if $y_2 \in S_1 \setminus S_2$, then $y_2 \in B$. So R together with the $y_1 - y_2$ part of $R_{j+1}R_{j+2} \cdots R_{n+j+1}$ forms an odd cycle. Hence it contains an odd circuit, and so R contains an odd bow. By (2.1), this bow connects two vertices in some R_{j+2}, \dots, R_n . This contradicts the fact that $j+2, \dots, n \notin J$. \square

Figure 1.1 gives a strongly t -perfect graph that contains a bad K_4 . So the implication in Theorem 3.1 cannot be reversed. However one has the following corollary.

COROLLARY 3.2. *For any graph G , the following are equivalent:*

- (3.38) (i) G contains no bad K_4 ;
 (ii) each subgraph of G is t -perfect;
 (iii) each subgraph of G is strongly t -perfect.

Proof. The implication (i) \Rightarrow (iii) follows from Theorem 3.1, while the implication (iii) \Rightarrow (ii) follows by the observations made in section 1.

The implication (ii) \Rightarrow (i) was proved by Barahona and Mahjoub [1]. It suffices to show that a bad K_4 is not t -perfect. Choose a smallest counterexample G . As G is t -perfect, $G \neq K_4$. If (1.4) does not hold, then G has a vertex v such that contracting the edges in $\delta(v)$ gives an odd K_4 -subdivision G' that again does not satisfy (1.4). As G' again is a t -perfect odd K_4 (as one easily checks), this contradicts the minimality of G . \square

Acknowledgment. I am very grateful to a referee for very carefully reading the manuscript and for giving several excellent suggestions for improving the presentation.

REFERENCES

- [1] F. BARAHONA AND A.R. MAHJOUR, *Compositions of graphs and polyhedra III: Graphs with no W_4 minor*, SIAM J. Discrete Math., 7 (1994), pp. 372–389.
- [2] M. BOULALA, J.-P. UHRY, *Polytope des indépendants d'un graphe série-parallèle*, Discrete Math., 27 (1979), pp. 225–243.
- [3] P.A. CATLIN, *Hajós' graph-coloring conjecture: Variations and counterexamples*, J. Combin. Theory Ser. B, 26 (1979), pp. 268–274.
- [4] J. FONLUPT AND J.P. UHRY, *Transformations which preserve perfectness and h -perfectness of graphs*, in Bonn Workshop on Combinatorial Optimization, Bonn, Germany, 1980, Ann. Discrete Math. 16, A. Bachem, M. Grötschel, and B. Korte, eds., North-Holland, Amsterdam, 1982, pp. 83–95.
- [5] J.F. GEELLEN AND B. GUENIN, *Packing odd-circuits in Eulerian graphs*, J. Combin. Theory Ser. B, to appear.
- [6] A.M.H. GERARDS, *A min-max relation for stable sets in graphs with no odd- K_4* , J. Combin. Theory Ser. B, 47 (1989), pp. 330–348.
- [7] A.M.H. GERARDS AND A. SCHRIJVER, *Matrices with the Edmonds-Johnson property*, Combinatorica, 6 (1986), pp. 365–379.
- [8] A.M.H. GERARDS AND F.B. SHEPHERD, *The graphs with all subgraphs t -perfect*, SIAM J. Discrete Math., 11 (1998), pp. 524–545.
- [9] H. HADWIGER, *Über eine Klassifikation der Streckenkomplexe*, Vierteljschr. Naturforsch. Ges. Zürich, 88 (1943), pp. 133–142.
- [10] N. SBIHI AND J.P. UHRY, *A class of h -perfect graphs*, Discrete Math., 51 (1984), pp. 191–205.

- [11] E.C. SEWELL AND L.E. TROTTER, JR., *Stability critical graphs and even subdivisions of K_4* , J. Combin. Theory Ser. B, 59 (1993), pp. 74–84.
- [12] P.D. SEYMOUR, *The matroids with the max-flow min-cut property*, J. Combin. Theory Ser. B, 23 (1977), pp. 189–222.
- [13] C. THOMASSEN, *Totally odd K_4 -subdivisions in 4-chromatic graphs*, Combinatorica, 21 (2001), pp. 417–443.
- [14] B. TOFT, *Problem 10*, in Recent Advances in Graph Theory (Proceedings Symposium Prague, 1974; M. Fiedler, ed.), Academia, Prague, 1975, pp. 543–544.
- [15] W. ZANG, *Proof of Toft's conjecture: Every graph containing no fully odd K_4 is 3-colorable*, J. Combin. Optim., 2 (1998), pp. 117–188.

HOW TO BE AN EFFICIENT SNOOP, OR THE PROBE COMPLEXITY OF QUORUM SYSTEMS*

DAVID PELEG[†] AND AVISHAI WOOL[‡]

Abstract. A *quorum system* is a collection of sets (quorums) every two of which intersect. Quorum systems have been used for many applications in the area of distributed systems, including mutual exclusion, data replication, and dissemination of information.

When the elements may fail, a user of a distributed protocol needs to quickly find a quorum all of whose elements are alive or evidence that no such quorum exists. This is done by *probing* the system elements, one at a time, to determine if they are alive or dead.

This paper studies the *probe complexity* $PC(\mathcal{S})$ of a quorum system \mathcal{S} , defined as the worst case number of probes required to find a live quorum or to show its nonexistence in \mathcal{S} , using the best probing strategy.

We show that for large classes of quorum systems, all n elements must be probed in the worst case. Such systems are called *evasive*. However, not all quorum systems are evasive; we demonstrate a system where $O(\log n)$ probes always suffice. Then we prove two lower bounds on the probe complexity in terms of the minimal quorum cardinality $c(\mathcal{S})$ and the number of minimal quorums $m(\mathcal{S})$. Finally, we show a universal probe strategy which never makes more than $c(\mathcal{S})^2 - c(\mathcal{S}) + 1$ probes; thus any system with $c(\mathcal{S}) \leq \sqrt{n}$ is nonevasive.

Key words. quorum systems, distributed computing, evasiveness, strong and simple games

AMS subject classifications. 68M14, 68Q25, 68R05, 91A10

PII. S0895480198343819

1. Introduction.

1.1. An illustrating scenario. The shareholders of the MegaBucks corporation need to vote on a decision with major implications. Due to a history of splits and merges, the voting structure is rather complicated, with many committees and subcommittees, often with a shareholder having a vote in many subcommittees. In game theory, such a voting structure is called a strong and simple game.

The reporter U.R. Nosey has the task of finding out whether the collective decision will be “yes” or “no.” He can do this by asking the voters, one by one, how they plan to vote (assuming nobody lies or changes their mind after talking to the reporter). Mr. Nosey can stop his snooping when he finds a collection of voters with the same opinion that together can force the outcome, i.e., when he finds a winning coalition all of whose members will vote the same way.

The main questions that we address in this paper are the following: How should Mr. Nosey choose the next voter to ask each time so he can finish his task with the smallest number of conversations? How many voters must he ask under the worst possible configuration of answers? In particular, must he ask all the voters?

*Received by the editors August 24, 1998; accepted for publication (in revised form) April 24, 2002; published electronically June 5, 2002. An extended abstract of this paper appeared in *Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing*, Philadelphia, PA, 1996, pp. 290–299.

<http://www.siam.org/journals/sidma/15-3/34381.html>

[†]Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel (peleg@wisdom.weizmann.ac.il). This author’s work was supported in part by grants from the Israel Science Foundation and from the Israel Ministry of Science and Art.

[‡]Department of Electrical Engineering—Systems, Tel Aviv University, Ramat Aviv 69778, Israel (yash@acm.org).

This game-theoretic scenario is analogous to the distributed systems scenario we have been dealing with all along. In the corresponding terminology, processors replace voters, winning coalitions are quorums, and the voting structure is the quorum system. The fact that the game is a strong and simple game is equivalent to the intersection property of a quorum system.

Any quorum-based distributed protocol must access, at some stage, a quorum all of whose processors are functioning. However, if processors may fail, then the protocol must *probe* the processors to determine if they are alive or dead, prior to using them. In the analogy with the snoop reporter's problem, the processors' live/dead states correspond to the voters' individual decisions. Like the reporter, the protocol needs to probe processors one by one until a live quorum is found or until it is certain that no such quorum exists.

Clearly, it is desirable to probe as few processors as possible, since the number of probes measures the communication complexity of the distributed protocol. Therefore, in quorum system language, we are interested in the following questions: What strategy should be used to probe a given quorum system efficiently? How many probes are necessary in the worst failure configuration? In particular, is it true that all processors must be probed in the worst case? The maximal number of probes needed to determine if a live quorum exists is what we call the *probe complexity* of a quorum system \mathcal{S} and is denoted by $\mathcal{PC}(\mathcal{S})$.

1.2. Related work. The rest of this paper uses the terminology of quorum systems. A good reference to game theory is [25]. Simple games, and their interpretations in reliability theory, are the subject of [29]. A discussion of the connection between strong and simple games and quorum systems can be found in [23].

Quorum systems serve as a basic tool providing a uniform and reliable way to achieve coordination between the processors and have been used in the study of problems such as *mutual exclusion* (cf. [30]), *data replication protocols* (cf. [6, 12, 36]), *distributed access control and signatures* (cf. [21]), and secure multiparty computation protocols (cf. [4]).

Many different quorum systems constructions appear in the literature. The simplest systems use *voting* to define the quorums [34, 10, 9]. Alternative constructions, which play a part in this paper, are found in [19, 7, 20, 1, 17, 13, 28].

Quorum systems, as tools for distributed protocols, were analyzed using various performance measures. The most widely studied measure is the availability (cf. [2, 26]). Other measures are the load [22] and load balancing [13]. A comprehensive analysis of all the above-mentioned quorum systems and some others can be found in [35].

The measure we call probe complexity is equivalent to the notion of the *argument complexity* of a boolean function, which is the maximum number of arguments of a boolean function f that must be tested in order to compute f . Aanderaa and Rosenberg conjectured that every nontrivial, monotone boolean function over n variables, describing a *graph property*, requires $\Omega(n)$ arguments to be tested in the worst case [32]. Karp conjectured that in fact every such property is *evasive*, i.e., requires that all n arguments be tested. The Aanderaa–Rosenberg conjecture was proved by Rivest and Vuillemin [31], who also showed that almost every boolean function is evasive for large n . Karp's stronger conjecture was later proved by [15].

To our knowledge, the probe complexity of quorum systems, or, equivalently, the argument complexity of boolean functions characterizing a quorum system, has not been studied before. In fact, most of the techniques of [31] and [15] are not applicable

in our case. This is since both proofs rely on the fact that a graph property P has a nice algebraic structure: the group of permutations of the k vertices acts transitively on the $n = \binom{k}{2}$ edges while preserving P . Boolean functions characterizing quorum systems rarely have such symmetry.

Following the appearance of our preliminary results in [27], Bazzi [3] introduced the related measure of cost-of-failures of a quorum system. This is the maximal number of probes that are needed per failure, which is essentially our probe complexity normalized by the number of failures that occurred.

1.3. New results. This paper addresses the question of how to quickly search for a live quorum in a distributed system when failures may occur, in the worst case model. Namely, we assume that an adversary, whose purpose is to force the user to make many probes, decides which elements fail.

After formalizing our model, we start with a discussion of evasiveness. We prove that large classes of quorum systems are evasive, including voting systems, crumbling walls, the finite projective plane, and compositions of these. However, and somewhat surprisingly, we show that not all quorum systems are evasive. We do this by demonstrating that the Nuc system of [7] requires only $O(\log n)$ probes in the worst case.

Next we prove two general lower bounds on the probe complexity of a quorum system in terms of combinatorial parameters of the system \mathcal{S} . We show that if the smallest quorum is of cardinality $c(\mathcal{S})$, then $\mathcal{PC}(\mathcal{S}) \geq 2c(\mathcal{S}) - 1$, and this bound is exactly tight for some examples. We also show that if \mathcal{S} has $m(\mathcal{S})$ minimal quorums,¹ then $\mathcal{PC}(\mathcal{S}) \geq \log_2(m(\mathcal{S})) + 1$.

After these essentially negative results, we describe a more positive result. We give a universal probing strategy and prove an upper bound on the number of probes it makes in the worst case. We show that if all the quorums are of the same cardinality c (a uniform quorum system), then at most $c^2 - c + 1$ probes always suffice. As a corollary we obtain that every uniform quorum system with $c(\mathcal{S}) \leq \sqrt{n}$ is *not* evasive.

The organization of this paper is as follows. In section 2 we introduce the definitions and notation of quorum systems. In section 3 we introduce the probe model of a quorum system. In section 4 we prove that large classes of quorum systems are evasive, and we show an example of a nonevasive system. The two lower bounds we prove on the probe complexity appear in section 5. The universal strategy and its analysis are in section 6. Finally, the topic of probe complexity presents a number of significant problems which are still unresolved, and in section 7 we list several open questions and directions for further research.

2. Preliminaries.

2.1. Basic definitions.

DEFINITION 2.1. A set system $\mathcal{S} = \{S_1, \dots, S_m\}$ is a collection of subsets $S_i \subseteq U$ of a finite universe U . A quorum system is a set system \mathcal{S} that has the following intersection property: $S \cap R \neq \emptyset$ for all $S, R \in \mathcal{S}$.

Alternatively, quorum systems are known as *intersecting set systems* or as *intersecting hypergraphs*. The sets of the system are called *quorums*. The number of elements in the underlying universe is denoted by $n = |U|$. The number of sets (quorums) in the set system \mathcal{S} is denoted by $m(\mathcal{S})$, and the cardinality of the smallest quorum in \mathcal{S} is denoted by $c(\mathcal{S}) = \min\{|S| : S \in \mathcal{S}\}$.

¹A quorum S (of any cardinality) is called minimal if all its proper subsets $R \subset S$ are nonquorums.

DEFINITION 2.2. Let \mathcal{S} be a quorum system. \mathcal{S} is s -uniform if $|S| = s$ for all $S \in \mathcal{S}$.

DEFINITION 2.3. A Coterie is a quorum system \mathcal{S} that has the minimality property: there are no $S, R \in \mathcal{S}$ such that $S \subset R$.

DEFINITION 2.4. Let \mathcal{R}, \mathcal{S} be coterie (over the same universe U). Then \mathcal{R} dominates \mathcal{S} , denoted $\mathcal{R} \succ \mathcal{S}$, if $\mathcal{R} \neq \mathcal{S}$ and for each $S \in \mathcal{S}$ there is $R \in \mathcal{R}$ such that $R \subseteq S$. A coterie \mathcal{S} is called dominated if there exists a coterie \mathcal{R} such that $\mathcal{R} \succ \mathcal{S}$. If no such coterie exists, then \mathcal{S} is nondominated (ND). Let NDC denote the class of all ND coterie.

ND coterie are the “best” quorum systems in that they have the highest availability [26] and lowest load [22]. In what follows all the quorum systems are ND unless otherwise noted.

DEFINITION 2.5. A set R is a transversal of a set system \mathcal{S} if $R \cap S \neq \emptyset$ for every $S \in \mathcal{S}$.

LEMMA 2.6 (see [9]). Let $\mathcal{S} \in \text{NDC}$, and let R be a transversal of \mathcal{S} . Then there exists a quorum $S \in \mathcal{S}$ such that $S \subseteq R$.

Given an ND quorum system \mathcal{S} , we find it useful to count the transversals according to their cardinalities and to use the following combinatorial lemma.

DEFINITION 2.7. Let $a_i^{\mathcal{S}}$ denote the number of size- i transversals of \mathcal{S} , i.e., the number of sets of size i that hit all the quorums of \mathcal{S} for $0 \leq i \leq n$:

$$a_i^{\mathcal{S}} = |\{X \in U : |X| = i \text{ and } \forall S \in \mathcal{S}, S \cap X \neq \emptyset\}|.$$

The vector $a^{\mathcal{S}} = (a_0^{\mathcal{S}}, \dots, a_n^{\mathcal{S}})$ is called the availability profile of \mathcal{S} .

LEMMA 2.8 (see [26]). Let $\mathcal{S} \in \text{NDC}$ be given. Then $a_i^{\mathcal{S}} + a_{n-i}^{\mathcal{S}} = \binom{n}{i}$ for $0 \leq i \leq n$.

An alternative view of a quorum system is that of a boolean function.

DEFINITION 2.9. Let \mathcal{S} be a quorum system. Let x_1, \dots, x_n be boolean variables corresponding to the elements of the universe. Then the characteristic function of \mathcal{S} is $f_{\mathcal{S}} : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by

$$f_{\mathcal{S}}(x_1, \dots, x_n) = \bigvee_{S \in \mathcal{S}} \bigwedge_{i \in S} x_i.$$

Clearly, $f_{\mathcal{S}}$ is monotone, and $f_{\mathcal{S}}(\mathbf{x}) = 1$ iff all the variables corresponding to some quorum have the value 1. Properties of characteristic functions of quorum systems are discussed extensively in [29, 14].

2.2. Examples. Let us illustrate the concept of quorum systems by giving some examples that play an important role in the results of this paper. The following constructions are all known to be ND coterie.

The majority system [34], denoted by Maj, is the collection of all sets of $\frac{n+1}{2}$ elements over a universe U when $n = |U|$ is odd.

The Wheel [13] contains $n - 1$ “spoke” quorums of the form $\{1, i\}$ for $i = 2, \dots, n$, and one “rim” quorum, $\{2, \dots, n\}$.

In the finite projective plane (FPP) system of [20], $n = t^2 + t + 1$ for t which is a power of a prime. The quorums are all of size $t + 1$ and correspond to the lines of the projective plane.

The crumbling walls are a family of quorum systems due to [28]. The elements of a wall are logically arranged in rows of varying widths. A quorum in a wall is the union of one full row and a representative from every row below the full row. The

Wheel is a crumbling wall with two rows of width 1 and $n-1$. The triangular (Triang) system [19, 7] is another crumbling wall, in which row i has width i .

In the Tree system [1] the elements are organized in a complete rooted binary tree. A quorum in the system is defined recursively to be either (i) the union of the root and a quorum in one of the two subtrees or (ii) the union of two quorums, one in each subtree.

In the HQC system [17], the elements are the leaves of a complete ternary tree. The internal nodes are 2-of-3 majority gates.

The nucleus (Nuc) system of [7] is built in two stages. First, consider a nucleus universe U_1 of size $2r-2$ for some $r > 1$ and add to \mathcal{S} all the subsets of U_1 of size r (call these “type A” quorums). Second, for each possible partition of U_1 into two disjoint sets T'_j, T''_j with $|T'_j| = |T''_j| = r-1$, add a new element x_j to the universe and add the sets $T'_j \cup \{x_j\}$ and $T''_j \cup \{x_j\}$ to \mathcal{S} . (These are “type B” quorums.)

3. The probe model. We assume that the elements (processors) of the system may occasionally fail. We assume that these failures are *crash* failures and that they are *detectable*. We also assume that the state of a processor does not change while the system is being probed; i.e., the processors are “fail-stop” [33]. We do not consider “lying” processors (Byzantine failures) or asynchronous communication with unbounded message delay.

When the protocol requires a user Alice to access a quorum, we assume that the configuration of failures is unknown to her. She can learn the configuration by *probing* the elements of the system one at a time (say by sending a message and waiting a timeout period for the reply). After probing element i , Alice knows if i is alive or dead.

Alice’s task is to find a live quorum, or a witness that no such quorum exists, with the minimal number of probes. Note that if no live quorum exists, then the set R of dead elements comprise a transversal of the system \mathcal{S} . However, by Lemma 2.6 it follows that, for an ND system \mathcal{S} , R must contain some quorum S as a subset, all of whose elements are dead. Therefore Alice’s stopping condition is symmetric for ND systems: find a quorum that contains only live elements or only dead elements.

We often refer to the live/dead state of the elements as a *coloring* of the universe U by calling a dead element “black” and a live element “white.” Therefore Alice’s task for an NDC system is as follows:

“Find a monochromatic quorum with the smallest number of probes.”

We allow Alice to use an *adaptive strategy* to decide which element to probe next, based on the results of all the previous probes. We do not consider probabilistic strategies; i.e., Alice cannot flip coins. Therefore every probe strategy can be described by a rooted binary tree, with labels on the nodes (see Figure 3.1). A tree node labeled i represents a probe of element $i \in U$, and the first probe is to the element appearing at the root. The two outgoing edges from a node correspond to the probe results: the left edge is followed when i is alive, and the right edge when i is dead. The tree leaves represent stopping states for Alice and are colored black or white according to whether a live (white) quorum was found or a dead (black) one. Additionally, we attach the names of the found monochromatic quorums to the leaves (the witness quorums).

We are interested in the worst case number of probes that are necessary to guarantee the finding of a monochromatic quorum. Hence we have the following definition.

DEFINITION 3.1. *Let $\mathcal{S} \in \text{NDC}$ be a quorum system. Then the probe complexity*

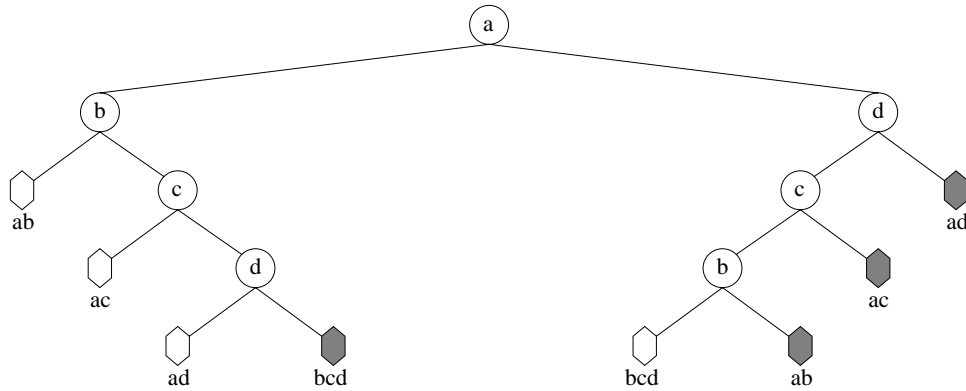


FIG. 3.1. A possible probe strategy for the Wheel_4 system: $\{\{ab\}, \{ac\}, \{ad\}, \{bcd\}\}$. Reaching a white hexagonal leaf indicates that a live quorum was found, and reaching a shaded one indicates a dead quorum. The names of the corresponding witness quorums appear below each leaf.

of \mathcal{S} is

$$\mathcal{PC}(\mathcal{S}) = \min_T \{depth(T)\},$$

where the minimum is taken over all possible probe strategy trees T , and the depth is the number of nodes on the longest path from the root to a leaf in T (not counting the leaf itself).

For example, the depth of the strategy shown in Figure 3.1 is 4, and it turns out that no strategy can do better, so $\mathcal{PC}(\text{Wheel}_4) = 4$. In fact, in what follows we show that $\mathcal{PC}(\text{Wheel}_n) = n$ for any universe size n . Such systems, which require all the elements to be probed (in the worst case) before a monochromatic quorum can be found, are especially important to us.

DEFINITION 3.2. Let \mathcal{S} be an ND quorum system over a universe of size n . If $\mathcal{PC}(\mathcal{S}) = n$, then we say that \mathcal{S} is evasive.

It is often useful to view Definition 3.1 as if Alice is playing a game against an adversary that controls the outcomes of the probes. The adversary knows Alice’s strategy and has an unbounded computational power. The adversary’s task is to force Alice to make as many probes as possible. Note that, since the adversary knows Alice’s strategy, it can search the (possibly exponential sized) tree and find the deepest leaf and then choose a failure configuration that forces Alice to reach it. However, sometimes we can give explicit adversary strategies that do not exhaustively search Alice’s strategy tree.

Evasiveness can be defined analogously for any boolean function f . Then a live element corresponds to a variable with value “1” and a dead element to a value “0.” A function f is evasive if all n inputs need to be tested before the function can be evaluated, in the worst case. Saying that a quorum system \mathcal{S} is evasive (as in Definition 3.2) is equivalent to saying that its characteristic function $f_{\mathcal{S}}$ is evasive as a boolean function (recall Definition 2.9).

4. Evasiveness. In this section we address the issue of evasiveness. Our starting point is the algebraic approach of [31], which we show to have limited usefulness in our case. Then in sections 4.2–4.5 we prove that large classes of quorum systems are evasive. Finally in section 4.6 we show that, surprisingly, there exist nontrivial quorum systems which are *not* evasive.

4.1. The algebraic approach. As part of their work on the evasiveness of graph properties, Rivest and Vuillemin [31] give in their Corollary 3.3 a sufficient condition for the evasiveness of a general monotone boolean function. Below we rephrase their result using our terminology to obtain a condition for quorum system evasiveness based on the availability profile (recall Definition 2.7).

PROPOSITION 4.1 (see [31]). *Let $a^{\mathcal{S}}$ be the availability profile of a quorum system $\mathcal{S} \in \text{NDC}$. If*

$$\sum_{i \text{ even}} a_i^{\mathcal{S}} \neq \sum_{i \text{ odd}} a_i^{\mathcal{S}},$$

then $\mathcal{PC}(\mathcal{S}) = n$; i.e., \mathcal{S} is evasive.

Example 4.2. The only FPP system [20] that is ND is the 7-point Fano plane [8]. For this system we have $a^{\text{FPP}} = (0, 0, 0, 7, 28, 21, 7, 1)$ by inspection, so the sum on the even indices is 35 while on the odd indices it is 29. Therefore by Proposition 4.1 the FPP system with $n = 7$ is evasive.

In their proof that almost all n -argument boolean functions f are evasive for large n , [31] in fact shows that the condition of Proposition 4.1 holds for all but an exponentially small fraction of the boolean functions. However, when we consider only the class NDC, the next proposition shows that Proposition 4.1 has limited usefulness.

PROPOSITION 4.3. *Let $\mathcal{S} \in \text{NDC}$ be over a universe of size $n = 2k$. Then*

$$\sum_{i \text{ even}} a_i^{\mathcal{S}} = \sum_{i \text{ odd}} a_i^{\mathcal{S}}.$$

Proof. Assume that k is odd. Note that, since n is even, if i is even, then so is $n - i$. Then using Lemma 2.8 and a combinatorial identity (cf. [16]) we obtain

$$\sum_{\substack{0 \leq i \leq n \\ i \text{ even}}} a_i^{\mathcal{S}} = \sum_{\substack{0 \leq i < k \\ i \text{ even}}} (a_i^{\mathcal{S}} + a_{n-i}^{\mathcal{S}}) = \sum_{\substack{0 \leq i < k \\ i \text{ even}}} \binom{n}{i} = \frac{1}{2} \sum_{\substack{0 \leq i \leq n \\ i \text{ even}}} \binom{n}{i} = 2^{n-2}.$$

However, a direct consequence of Lemma 2.8 is that $\sum a_i^{\mathcal{S}} = 2^{n-1}$. Therefore since the sum over the even indices is 2^{n-2} then so is the sum over the odd indices. The case where k is even is handled analogously. \square

4.2. The adversary approach. An alternative method of proving that a quorum system is evasive is by giving an explicit strategy for an *oblivious* adversary that forces the user Alice to probe all n elements. An oblivious adversary is weaker than the adversary of Definition 3.1: it does not know Alice’s strategy.

DEFINITION 4.4. *An oblivious adversary strategy is a procedure which computes the answer to a probe of any element $i \in U$, based only on the history of probes and answers.*

DEFINITION 4.5. *A quorum system \mathcal{S} is called obliviously evasive if there exists an oblivious adversary strategy A which forces the user Alice to probe all n elements for any probing strategy she uses.*

An unbounded adversary, which knows Alice’s strategy, can certainly simulate any oblivious adversary strategy. Therefore if a system \mathcal{S} is obliviously evasive, then it is also evasive in the regular sense.

4.3. Composite systems. Our next goal is to prove Theorem 4.7, which allows us to prove the evasiveness of quorum systems that have a composite structure. For this purpose we use the characteristic function f_S of a quorum system, with the interpretation that $x_i = 1$ for a live element i and $x_i = 0$ otherwise. As we pointed out before, the evasiveness of the characteristic function is equivalent to the evasiveness of the quorum system.

LEMMA 4.6. *Let f be an obviously evasive function. Then there exists an oblivious adversary strategy $A(\alpha)$ which ensures that f evaluates to $\alpha \in \{0, 1\}$, and the decision between 0 and 1 is made after forcing Alice to make $n - 1$ probes.*

Proof. By definition there exists an oblivious adversary strategy B which forces Alice to make n probes. Note that after making $n - 1$ probes against B Alice cannot stop yet. Therefore there exist two configurations \mathbf{x}_0 and \mathbf{x}_1 that differ only in the value of the unprobed element i and that agree with the probe results on all other elements such that $f(\mathbf{x}_0) = 0$ and $f(\mathbf{x}_1) = 1$.

The strategy $A(\alpha)$ is the following. For the first $n - 1$ probes return the answer given by strategy B . Suppose the answer given by B to the n th probe is b , which causes f to evaluate to β . If $\alpha = \beta$, then return b ; otherwise, return $1 - b$.

If β equals the desired output α , then the correctness of A is obvious. Otherwise, flipping the bit b changes the resulting configuration from \mathbf{x}_0 to \mathbf{x}_1 , say, which in turn changes the output to α . \square

THEOREM 4.7. *If $f(x_1, \dots, x_t)$ is an obviously evasive boolean function, and $\{g^j(y_1^j, \dots, y_{n_j}^j)\}_{1 \leq j \leq t}$ is a family of t obviously evasive functions on n_j variables, respectively, then the function*

$$f \circ \mathbf{g} = f(g^1(y_1^1, \dots, y_{n_1}^1), \dots, g^t(y_1^t, \dots, y_{n_t}^t))$$

is obviously evasive on $n = \sum_{1 \leq j \leq t} n_j$ variables.

Proof. Since f and $\{g^j\}_{1 \leq j \leq t}$ are all obviously evasive, the adversary has strategies A_f and A_{g^j} that force Alice to probe all the inputs in each function separately. The composite adversary strategy is the following. When Alice probes an input y which belongs to some g^j in $f \circ \mathbf{g}$ then we have the following:

- If less than n_j of g^j 's inputs were probed so far, return the answer given by A_{g^j} to the probe.
- If this is the n_j th probe of an input of g^j , then first activate the strategy A_f to determine the answer α for a probe of f 's input x_j . Then activate $A_{g^j}(\alpha)$ and return the value that forces g^j to evaluate to α . (This can be done by Lemma 4.6.)

Since f is obviously evasive, the use of strategy A_f ensures that the value of $f \circ \mathbf{g}$ remains undetermined until all the g^j functions are evaluated. (The evaluation of a function g^j is treated as a probe of the variable x_j of f .) Since the inputs sets of the g^j functions are disjoint, it is clear that all n_j inputs of each function must be probed before the value of g^j can be determined. \square

Next we use Theorem 4.7 to prove (in Corollary 4.10) that the Tree and HQC systems are evasive. Proposition 4.9 serves as a building block for the proof.

DEFINITION 4.8. *A threshold “ k -of- n ” function is a boolean function on n variables that attains the value 1 iff at least k of its inputs have the value 1.*

PROPOSITION 4.9. *Every threshold “ k -of- n ” function is evasive.*

Proof. An adversary strategy $A(\alpha)$ which forces the user Alice to probe all n inputs is the following: Answer the first $k - 1$ probes by “1.” Answer the next $n - k$ probes by “0.” Answer the n th probe by α . \square

COROLLARY 4.10. *The Tree [1] and HQC [17] quorum systems are evasive.*

Proof. By Proposition 4.9 the 2-of-3 majority function is evasive. The HQC system is a complete ternary tree of 2-of-3 majorities, so by induction on the tree height and using Theorem 4.7 at each level it follows that HQC is evasive. A description of the Tree system as another tree of 2-of-3 majorities appears in [14], so a similar inductive proof shows its evasiveness. \square

Remark. In fact, [14, 18] show that any NDC can be decomposed into a tree of 2-of-3 majorities. The Tree and HQC systems have decompositions that are *read-once*; i.e., each variable is input to a single 2-of-3 majority, so Theorem 4.7 can be used. However, in general, the decomposition is not read-once, so Theorem 4.7 cannot be applied.

4.4. Crumbling wall systems. Here we show another application of Theorem 4.7, which proves that the class of crumbling walls consists of evasive quorum systems.

PROPOSITION 4.11. *The Wheel quorum system is evasive.*

Proof. An adversary strategy $A(\alpha)$ which forces the user Alice to probe all n inputs is the following: If a rim element is probed during the first $n - 2$ probes answer “0.” If probe $n - 1$ is to a rim element, and so were all the previous probes, then answer “1”; otherwise, answer “0.” If the hub is probed during the first $n - 1$ probes answer “1.” Answer the n th probe by α .

If Alice probes the hub among her first $n - 1$ probes, she will reach the n th probe, since the hub has value 1 and every known rim element has 0. Otherwise, she probes all $n - 1$ rim element first to discover that they do not all have 0, so she must probe the hub as well. \square

THEOREM 4.12. *Every crumbling wall quorum system is evasive.*

Proof. Informally, the adversary strategy is a variant of the following strategy: For any row i with n_i elements, answer the first $n_i - 1$ queries with “0”; answer the n_i th query with “1.” It is not hard to see that this strategy forces Alice to make n probes. However, as stated the outcome is always “1,” and so we need to modify strategy so that a “0” outcome is also possible. We now give a formal proof that this modified strategy is indeed an oblivious adversary strategy.

Consider a wall W on $d > 1$ rows, whose bottom row contains the elements u_1, \dots, u_{n_d} , and let g_d be its characteristic function. Denote the characteristic function of the crumbling subwall on the top $d - 1$ rows by g_{d-1} . Let $f(x_0, x_1, \dots, x_{n_d})$ denote the characteristic function of the Wheel system on $n_d + 1$ variables, with variable x_0 corresponding to the hub. Then it is easy to see that the wall W can be decomposed into a Wheel whose hub is replaced by the top $d - 1$ row subwall. Formally, $g_d = f(g_{d-1}, u_1, \dots, u_{n_d})$. Thus we obtain a recursive decomposition of a crumbling wall using building blocks which are all Wheel systems and singletons on disjoint sets of elements. The Wheel system is evasive by Proposition 4.11, and singletons are trivially evasive, so we can apply Theorem 4.7 inductively and we are done. For the base of the induction, note that a crumbling wall with a single row is an n -of- n threshold system, so it is evasive by Proposition 4.9. \square

4.5. Voting systems. Via the following definitions and lemmas we prove (in Theorem 4.18) that every quorum system defined by voting, which has no dummy elements, is evasive.

Notation. For a vector $v \in \mathbb{Z}^n$ and a set $S \subseteq U$, let $v(S) = \sum_{i \in S} v_i$.

DEFINITION 4.13. Let $v \in \mathbb{Z}^n$ and an integer threshold T be given. The voting system $(v; T)$ is the collection of all the minimal sets $S \subseteq U$ such that $v(S) \geq T$:

$$(v; T) = \{S \subseteq U : v(S) \geq T \text{ and } \forall u \in S, v(S \setminus \{u\}) < T\}.$$

Remark. A voting system is a quorum system (has the intersection property) iff the threshold $T > v(U)/2$. We need the more general definition for the proof of Theorem 4.18. However, with slight abuse of terminology we still refer to the sets of $(v; T)$ as “quorums.”

The voting system with weights $(4, 4, 4, 1)$ and threshold 7 is not evasive, since there is never any need to probe the element with weight 1. To avoid such trivialities we add the following definition.

DEFINITION 4.14. Let $(v; T)$ be a voting system $v(U) = V$. An element $i \in U$ is a dummy if it does not belong to any (minimal) quorum, or, formally, $i \notin \cup\{S : S \in (v; T)\}$.

DEFINITION 4.15. Let $(v; T)$ be a voting system with $v(U) = V$. A critical partition for i is a partition $W|B$ of $U \setminus \{i\}$ into two sets W and B such that

- (1) $v(W) < T$ and $v(W \cup \{i\}) \geq T$,
- (2) $v(B) \leq V - T$ and $v(B \cup \{i\}) > V - T$.

LEMMA 4.16. Let $(v; T)$ be a voting system with $v(U) = V$. An element $i \in U$ is not a dummy in $(v; T)$ iff there exists a critical partition for i .

Proof. (\Rightarrow) Assume that i is not a dummy. Then there exists a (minimal) quorum $S \in (v; T)$ such that $i \in S$. For this S , take $W = S \setminus \{i\}$ and $B = U \setminus S$. To prove (1), note that $v(W \cup \{i\}) = v(S) \geq T$ by definition, and $v(W) = v(S \setminus \{i\}) < T$ by the minimality of S . Now (1) implies (2), since $v(B) = v(U \setminus S) = V - v(S) \leq V - T$ and $v(B \cup \{i\}) = v(U \setminus W) = V - v(W) > V - T$.

(\Leftarrow) Assume there exists a partition $W|B$ of $U \setminus \{i\}$ obeying (1) and (2). Take $R = W \cup \{i\}$. Then $v(R) = v(W \cup \{i\}) \geq T$ by (1). Now discard elements from R until it becomes a minimal set S for which $v(S) \geq T$ still holds. Then we claim that $i \in S$: Assume that i was discarded, then $v(S) \leq v(R \setminus \{i\}) = v(W) < T$, in contradiction to the definition of S . Hence $S \in (v; T)$ and $i \in S$, so i is not a dummy. \square

LEMMA 4.17. Let j be an element with the minimal weight v_j . If j is not a dummy in the voting system $(v; T)$, then $(v; T)$ is dummy-free; i.e., no element $i \in U$ is a dummy.

Proof. Since j is not a dummy, there exists a minimal $S \in (v; T)$ such that $j \in S$. Consider some other element i . If $i \in S$ we are done, so assume otherwise. Take the set $R = S \setminus \{j\} \cup \{i\}$. Then $v(R) = v(S) - v_j + v_i \geq v(S) \geq T$, since v_j is the minimal weight. Now discard elements from R until a minimal set R' is obtained for which $v(R') \geq T$ still holds. We claim that $i \in R'$: otherwise, $v(R') \leq v(R \setminus \{i\}) = v(S \setminus \{j\}) < T$ by the minimality of S , in contradiction to the definition of R' . Hence $i \in R' \in (v; T)$ and i is not a dummy. \square

THEOREM 4.18. Every dummy-free voting system is evasive.

Proof. For a voting system $(v; T)$, and a probe of element $i \in U$, the adversary uses the following oblivious strategy:

1. Let j be a minimal weight element in $U \setminus \{i\}$.
2. Find a critical partition $W|B$ for j .
3. If the probed element $i \in W$, then answer “white”; otherwise, answer “black.”

Since $(v; T)$ is dummy-free, and in particular j is not a dummy, by Lemma 4.16 a critical partition $W|B$ can be found in step 2 for this j . Therefore the adversary's strategy is well defined.

After answering the probe of element i we obtain a new voting system $(v'; T')$ on $U \setminus \{i\}$, with $v' = (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$, and either $T' = T - v_i$ (if the answer was “white”) or $T' = T$ (if the answer was “black”). Let $V' = v'(U \setminus \{i\}) = V - v_i$. Alice can stop probing in one of two cases:

- $T' \leq 0$. Then i completes a white quorum, so i must have been in W .
- $T' > V'$. Then no white quorum can be found, so i must have been in B .

Suppose that $i \in W$. However, $W|B$ is a critical partition for j , which was not probed yet and is outside W . So $T' = T - v_i \geq T - v(W) > 0$ by condition (1) of Lemma 4.16, and hence Alice cannot stop after a “white” answer. If $i \in B$, then, by condition (2) of Lemma 4.16, we get $V' = V - v_i \geq V - v(B) \geq V - (V - T) = T'$, so Alice cannot stop after a “black” answer either.

We still need to show that $(v'; T')$ is dummy-free, and then the proof is complete by induction on the universe size n . Assume that $i \in W$. Since $W|B$ was critical for j , after answering the probe on i the partition $W \setminus \{i\}|B$ is clearly still critical for j (in the universe $U \setminus \{i\}$). A similar situation occurs when $i \in B$. However, j is a minimal weight element in $(v'; T')$, so by Lemma 4.17 it follows that the resulting voting system $(v'; T')$ is also dummy-free, and we are done. \square

Remark. Finding the critical partition in step 2 is an NP-hard problem, but we assumed that the adversary has unbounded power.

4.6. Nonevasive examples. All the examples we have seen so far are of evasive quorum systems. Furthermore, in [31] it is shown that almost every boolean function on n variables is evasive for large n . Therefore it is reasonable to expect that, in a class of functions that has a “nice” structure, all the functions are evasive. This indeed is the case for graph-property functions, as shown by [31, 15]. However, for the class of ND quorum systems this is *not* the case. Below we show an ND uniform quorum system that has no dummy elements (i.e., every element belongs to some minimal quorum), which is not evasive.

Consider the Nuc system of [7] described in section 2.2. All its quorums are of size $c(\text{Nuc}) = r$, and it has $n = 2r - 2 + \frac{1}{2} \binom{2r-2}{r-1}$ elements, so $c(\text{Nuc}) \approx \frac{1}{2} \log_2 n$. The next proposition shows that in the Nuc system, $O(\log n)$ probes always suffice.

PROPOSITION 4.19. *$2r - 1$ probes are always sufficient to find a monochromatic quorum in the Nuc system.*

Proof. Consider the following strategy. First probe the $2r - 2$ elements of the nucleus. If at some stage r of these elements are found to have the same color—stop; a monochromatic quorum (of type A) was found.

The only configurations that require more probes are those in which the nucleus is partitioned into two sets of size $r - 1$, T and T' , of black and white elements, respectively. However, for every such partition there exists a unique element y outside the nucleus such that both $T \cup \{y\}$ and $T' \cup \{y\}$ are type B quorums. Therefore after probing this element y a monochromatic quorum will certainly be found. \square

We shall see in section 6 that every uniform quorum system with $c(\mathcal{S}) \leq \sqrt{n}$ is nonevasive. However, is this condition sufficient? In fact, the question “do all nonevasive quorum systems have $c(\mathcal{S}) \leq \sqrt{n}$?” was left open in [27]. Here we answer this question in the negative, by showing a family of uniform ND quorum systems, with $c(\mathcal{S})$ ranging from $O(\log n)$ up to $n/2$, which are all nonevasive.

These systems are modifications of the Nuc system and are parameterized by a

number k . The $\text{Nuc}(k)$ system has $2r - 2$ nucleus elements and k satellite elements b_0, \dots, b_{k-1} . All the sets of r nucleus elements are quorums (of type A) of $\text{Nuc}(k)$. As for quorums of type B, we have the following rule. Enumerate the ways to partition the nucleus into two disjoint sets $T_j; T'_j$ with $|T_j| = |T'_j| = r - 1$, using an index $j = 1, \dots, \frac{1}{2} \binom{2r-2}{r-1}$. Then, for partition number j , the two corresponding type B quorums are $T_j \cup \{b_j \pmod k\}$ and $T'_j \cup \{b_j \pmod k\}$. In other words, $\text{Nuc}(k)$ is a Nuc system in which the satellite elements may appear in more than one pair of type B quorums.

FACT 4.20. *$\text{Nuc}(k)$ is an ND, dummy-free, r -uniform quorum system for any $1 \leq k \leq \frac{1}{2} \binom{2r-2}{r-1}$. It has $n = 2r - 2 + k$ elements and $m(\text{Nuc}(k)) = \frac{1}{2} \binom{2r}{r}$ quorums.*

Remark. When $k = \frac{1}{2} \binom{2r-2}{r-1}$ then $\text{Nuc}(k) \equiv \text{Nuc}$, so its minimal quorum size is $c(\text{Nuc}(k)) = r \approx \frac{1}{2} \log n$ as before. When $k = 1$ then $\text{Nuc}(k)$ is precisely the Maj system over $n = 2r - 1$ elements; i.e., $c(\text{Nuc}(1)) = (n + 1)/2$.

PROPOSITION 4.21. $\mathcal{PC}(\text{Nuc}(k)) \leq 2r - 1$ for all $k \geq 1$.

Proof. The proof is identical to Proposition 4.19. \square

COROLLARY 4.22. $\text{Nuc}(k)$ is nonevasive for all $k \geq 2$.

5. Lower bounds. In this section we prove two lower bounds on the probe complexity, in terms of the smallest quorum size $c(\mathcal{S})$ and the number of quorums $m(\mathcal{S})$.

Notation. For a set R let \mathbf{x}_R denote the configuration in which the elements of R are white and all others are black.

PROPOSITION 5.1. $\mathcal{PC}(\mathcal{S}) \geq 2c(\mathcal{S}) - 1$ for any $\mathcal{S} \in \text{NDC}$.

Proof. First note that every correct strategy must probe at least $c(\mathcal{S})$ elements before stopping, regardless of the probe results, simply in order to probe all the elements of at least one quorum. Therefore the top $c(\mathcal{S})$ levels of any probe strategy tree are complete (see Figure 3.1).

Consider such a tree T , and consider L , its leftmost path from the root. By the above argument, L is at least $c(\mathcal{S})$ probes long. Let W be the set of elements labeling the top $c(\mathcal{S}) - 1$ nodes in L . There must exist a quorum $B \in \mathcal{S}$ such that $B \cap W = \emptyset$: otherwise, W is a transversal, which would imply that W contains a quorum by Lemma 2.6, contradicting the minimality of $c(\mathcal{S})$.

Now consider the configuration \mathbf{x}_W . On such a configuration, a user Alice using strategy T first probes all $c(\mathcal{S}) - 1$ elements of W and makes $c(\mathcal{S}) - 1$ left turns in her descent in the tree. At this point, no black element is encountered yet. However, the final decision must be black, since the quorum B is all black, so Alice must probe at least $c(\mathcal{S})$ more elements before reaching a black leaf. \square

Remark. Equality holds in Proposition 5.1 in the following cases:

- In the Maj system, $c(\text{Maj}) = \frac{n+1}{2}$, and by Proposition 4.9 Maj is evasive, so $\mathcal{PC}(\text{Maj}) = n$.
- In the Nuc system with a nucleus of size $2r - 2$ and $c(\text{Nuc}) = r$, Proposition 4.19 shows that $\mathcal{PC}(\text{Nuc}) \leq 2r - 1$, so Proposition 5.1 proves that in fact $\mathcal{PC}(\text{Nuc}) = 2r - 1$.

PROPOSITION 5.2. $\mathcal{PC}(\mathcal{S}) \geq \log_2(m(\mathcal{S})) + 1$ for any $\mathcal{S} \in \text{NDC}$.

Proof. Consider some probe strategy tree T . For each quorum $S \in \mathcal{S}$, let $\psi(S)$ be the (white) leaf in T which is reached when probing the configuration \mathbf{x}_S .

CLAIM 5.3. *Let $S, R \in \mathcal{S}$. If $S \neq R$, then $\psi(S) \neq \psi(R)$.*

Proof of the claim. Since $\mathcal{S} \in \text{NDC}$ it follows that $S \not\subseteq R$ and $R \not\subseteq S$. Let v be the first element in $(S \setminus R) \cup (R \setminus S)$ to be probed when the configuration is \mathbf{x}_S . Clearly, v is the first such element probed when the configuration is \mathbf{x}_R as well. Assume w.l.o.g.

that $v \in S$. Then the path to $\psi(S)$ makes a left turn at v , since v is white in \mathbf{x}_S , but the path to $\psi(R)$ turns right at v , so $\psi(S) \neq \psi(R)$.

Claim 5.3 shows that ψ is a one-to-one mapping of quorums to white leaves of T ; thus T has at least $m(\mathcal{S})$ white leaves. By swapping the roles of black and white and repeating the argument we obtain that T has at least $m(\mathcal{S})$ black leaves as well. Hence the depth of T is $\geq \log_2(2m(\mathcal{S}))$, which completes the proof of Proposition 5.2. \square

Remarks. This lower bound is tight (up to additive constants) for the Maj and Nuc systems and is trivially exact for the singleton system. The bound is sometimes better than that of Proposition 5.1, as the following examples show.

- In the Tree system [1], $c(\text{Tree}) \approx \log n$ and $m(\text{Tree}) \approx 2^{n/2}$, so Proposition 5.2 gives a linear lower bound of $\mathcal{PC}(\text{Tree}) \geq n/2$, much better than that of Proposition 5.1 but still short of the truth which is $\mathcal{PC}(\text{Tree}) = n$ by Corollary 4.10.
- The Triang system [19] is uniform with $c(\text{Triang}) \approx \sqrt{2n}$ and $m(\text{Triang}) = \Omega((\sqrt{n})!)$. Thus Proposition 5.2 gives $\mathcal{PC}(\text{Triang}) \geq \Omega(\sqrt{n} \log n)$, which is better than the bound of Proposition 5.1 by a logarithmic factor but far from the true value $\mathcal{PC}(\text{Triang}) = n$ shown by Theorem 4.12 (since the Triang is a crumbling wall).

6. A universal probing strategy. In this section we give a universal probing strategy (see Figure 6.1) that works for any ND quorum system. We prove that $c^2 - c + 1$ probes always suffice for a c -uniform ND system when Alice uses this strategy. As a corollary we prove that any c -uniform ND quorum system with $c \leq \sqrt{n}$ is *nonevasive*.

Two probing strategies were described in [27] and [35], called the “alternating color” strategy and the “white” strategy. Both of these are special cases of the universal strategy, in which additional rules dictate some of the choices made. Thus the unified treatment here is more general and implies the results stated in [27, 35].

Moreover, as noted by Nielsen [24], the alternating color strategy of [27] is not well defined for dominated quorum systems. In contrast, the universal strategy we present here is well defined for all quorum systems (dominated or not), and as such it is a marked improvement over our earlier results.

The above-mentioned white strategy is similar to a procedure that was used in a very different context in [5], as part of the argument for proving that if $P \neq NP \cap co-NP$, then $P \neq NP \cap co-NP$ with a generic oracle. The exposition in [5] treats infinite languages and thus does not include the combinatorial analysis of the number of probes that we have here.

We need the following technical definition for the description of the strategy.

DEFINITION 6.1. *During the probing procedure, an element’s color is either white, black, or unknown. A quorum $S \in \mathcal{S}$ is a white candidate (respectively, black candidate) if the colors of its elements are not all known and it has no black (respectively, white) elements.*

The strategy works in rounds. In every round the following steps are performed:

1. Pick candidate quorum S (either white or black) and probe all its unknown elements.
 2. If a monochromatic quorum is found, stop.
-

FIG. 6.1. *The universal strategy.*

Remarks.

- The strategy probes all the elements of the candidate quorum S even if it becomes clear that S cannot be the solution. For instance, if S was a white candidate at the beginning of a round, then the strategy continues to probe the elements of S even after black elements are encountered.
- The strategy stops if *any* monochromatic quorum is found. Its color may well be different from that of the candidate picked for this round.
- A bichromatic quorum, which was discovered to have both white and black elements in previous rounds, is not a candidate any more.

LEMMA 6.2. *Let $S \in \text{NDC}$ be given. If the universal strategy has not stopped by the end of round r , then both a white candidate and a black candidate still exist at the beginning of round $r + 1$.*

Proof. Assume to the contrary that the strategy reaches round $r + 1$ and a white candidate cannot be found. Then, by definition, every quorum has a (known) black element. Hence the set of black elements B is a transversal, and by Lemma 2.6 there exists some quorum $R \in \mathcal{S}$ for which $R \subseteq B$. However, this R is a monochromatic quorum, all of whose elements were probed. Therefore the strategy should have stopped after round r or earlier, in contradiction to the assumption that round $r + 1$ was reached. The case of a missing black candidate is identical. \square

Remark. Lemma 6.2 is incorrect for dominated systems. For example, consider the Star system $\{\{1, i\} : i = 2, \dots, n\}$. If after the first round it turns out that element 1 is white and element 2 is black, then no black candidate quorum can be found for round 2.

DEFINITION 6.3. *Let S_1, S_2, \dots, S_r be the candidate quorums picked in the first r rounds. Let $B_i \subseteq S_i$ be the set of black elements in S_i (the black part), and let $W_i \subseteq S_i$ be the white part for $1 \leq i \leq r$.*

LEMMA 6.4. *Assume that the strategy has not stopped by the end of round r . Let I_W and I_B be the sets of indices of the white and black candidates in the first r rounds, respectively. Then the black parts of the white candidates $\{B_i : i \in I_W\}$ are nonempty, disjoint sets and similarly for the white parts of the black candidates $\{W_j : j \in I_B\}$.*

Proof. If $B_i = \emptyset$ for some $i \in I_W$, then S_i is all white and the strategy should have stopped in round i , in contradiction to the assumption that a monochromatic quorum was not found up to round r .

Consider some $i \in I_W$. Note that S_i was a white candidate in round i , so at the beginning of the round all its known elements were white. Therefore $S_k \cap S_i \subseteq W_i$ for all $k < i$; thus S_i 's black part B_i is disjoint from every previous candidate S_k . In particular it is disjoint from the black part of every previous white candidate. The proof for the white parts of black candidates is analogous. \square

Remark. The quorum S_1 picked in round 1 is a black candidate and a white candidate simultaneously since all its elements' colors are unknown. All the subsequent quorums S_i are either white or black candidates, but not both, since $S_i \cap S_1 \neq \emptyset$ and the colors of all S_1 's elements are known.

DEFINITION 6.5. *Let w_r and b_r denote the numbers of white and black candidate quorums picked in the first r rounds, respectively.*

LEMMA 6.6. *Assume the strategy has not stopped by the end of round r . Then*

- *if S_{r+1} is a white candidate, then the colors of at least b_r of its elements are known (to be white);*
- *if S_{r+1} is a black candidate, then the colors of at least w_r of its elements are*

known (to be black).

Proof. Assume S_{r+1} is a white candidate. This S_{r+1} intersects each of the b_r previous black candidates, so the intersections must be in the black candidates' white parts. However, the white parts of the black candidates are nonempty and disjoint by Lemma 6.4. Therefore S_{r+1} has at least b_r elements whose color is known (to be white) at the beginning of round $r + 1$. The case of a black candidate is analogous. \square

PROPOSITION 6.7. *Let $\mathcal{S} \in \text{NDC}$ be c -uniform. Then the universal strategy stops after probing at most c white candidates and at most c black candidates.*

Proof. To obtain a contradiction, assume that $b_r = c$ black candidates were probed by the end of round r , but the strategy had not stopped yet. Then by Lemma 6.2 a white candidate W still exists. By Lemma 6.6, c of W 's elements are known to be white. However, $|W| = c$; thus W is already known to be monochromatic, in contradiction to the assumption that the strategy had not stopped. The argument for c white candidates is analogous. \square

A direct application of Proposition 6.7 gives an upper bound of $\mathcal{PC}(\mathcal{S}) \leq 2c^2$. However, $2c^2$ is quite a rough estimate. A more careful analysis allows us to prove the tight bound of the next theorem.

THEOREM 6.8. *Let $\mathcal{S} \in \text{NDC}$ be c -uniform. Then $\mathcal{PC}(\mathcal{S}) \leq c^2 - c + 1$.*

Proof. Let P_i denote the aggregate number of probed elements by the end of round i , and let w_i and b_i be as in Definition 6.5. We prove that the following invariant holds.

CLAIM 6.9. $P_i + (c - w_i)(c - b_i) \leq c^2 - c + 1$ for all rounds $i \geq 1$.

Proof. The proof is by induction on i . For the induction base, recall that the quorum picked in round 1 is both a white candidate and a black candidate, so $w_1 = b_1 = 1$, and since \mathcal{S} is c -uniform we have $P_1 = c$. So for $i = 1$ the invariant holds (with equality).

Now we assume the invariant holds for i and prove it holds for $i+1$. If a monochromatic quorum was found in round i , then the strategy stops and we are done. Otherwise, assume that the picked candidate S_{i+1} is white. By Lemma 6.6 we see that S_{i+1} has at least b_i elements whose color is known (to be white) at the beginning of round $i + 1$. Hence at most $c - b_i$ elements are probed in round $i + 1$ and

$$P_{i+1} \leq P_i + (c - b_i).$$

As remarked after Lemma 6.4, since S_{i+1} is a white candidate it cannot be a black candidate simultaneously. So $w_{i+1} = w_i + 1$ and $b_{i+1} = b_i$. Using the induction hypothesis we obtain that

$$\begin{aligned} P_{i+1} + (c - w_{i+1})(c - b_{i+1}) &\leq P_i + (c - b_i) + (c - w_i - 1)(c - b_i) \\ &= P_i + (c - w_i)(c - b_i) \\ &\leq c^2 - c + 1, \end{aligned}$$

and the invariant holds. The proof is analogous if S_{i+1} is a black candidate. This concludes the proof of Claim 6.9. \square

By Proposition 6.7 we have that the strategy stops after some $r \leq 2c$ rounds, at which time $w_r \leq c$ and $b_r \leq c$. For this r we have from the invariant of Claim 6.9 that

$$P_r \leq c^2 - c + 1 - (c - w_r)(c - b_r) \leq c^2 - c + 1,$$

and thus $\mathcal{PC}(\mathcal{S}) \leq c^2 - c + 1$. \square

COROLLARY 6.10. *Let $\mathcal{S} \in \text{NDC}$ be c -uniform. If $c \leq \sqrt{n}$, then \mathcal{S} is nonevasive.*

Remarks.

- Theorem 6.8 is exactly tight for the 7-element FPP system: every FPP with quorums of size c has $n = c^2 - c + 1$, and the 7-element system (the only ND one) is evasive by Example 4.2.
- Corollary 6.10 is a sufficient condition for nonevasiveness, but it is not necessary. The $\text{Nuc}(k)$ systems of section 4.6 are all c -uniform ND quorum systems which are nonevasive, but some of them have $c(\mathcal{S}) > \sqrt{n}$. In fact, for the $\text{Nuc}(k)$ systems Theorem 6.8 is not tight; the bound is $\approx c^2$, while $\approx 2c$ probes suffice by Proposition 4.19.

7. Concluding remarks and open questions. To the best of our knowledge, the question of how to search for a live quorum has not been addressed before in the context of distributed systems. We have demonstrated that the question is not a trivial one, especially when the system is defined by a combinatorial construction (rather than by voting). We believe that finding a good answer, in the form of a probing strategy and an analysis showing that it behaves “well,” is an important and interesting goal. Here we list some of the related open problems we are interested in.

- Perhaps the most interesting problem, from a practical point of view, is the *average case* analysis of probing strategies, i.e., when the configuration of failures is not determined by a malicious adversary but is chosen probabilistically. Our initial results in this direction provided some evidence that the behavior is qualitatively different from the worst case. For instance, the Wheel system is evasive, but there is a trivial strategy for which the average number of probes is ≈ 3 for any universe size n . This direction was studied further in [11], which presented upper and lower bounds for the deterministic average case probe complexity of quorum systems in some classes of ND coterie, including majority, crumbling walls, Tree, Wheel and hierarchical quorum systems.

A related problem concerns the probe complexity of *randomized* algorithms. This direction was also studied in [11], where it is shown that randomized algorithms may in many cases enjoy improved probe complexity in the worst case model compared to that achieved by deterministic ones.

- The universal strategy offers a large degree of freedom in choosing the candidate quorums—can this be used? An obvious rule would be to choose the candidate with the smallest number of elements whose color is unknown—does this (provably) help?
- Give a good probing strategy for nonuniform quorum systems. Note that our analysis of the universal probing strategy is essentially a “competitive analysis” with a competitive ratio of $c - 1 + 1/c$ for uniform systems. However, for nonuniform systems we must replace c with c_{\max} , the maximal quorum cardinality, and in nonuniform systems typically $c_{\max} = \Omega(n)$.
- Everyday intuition tells us to probe the elements according to their relative influence. Can game-theory measures of influence such as the Shapley value or the Banzhaf index be used to devise a provably good strategy? Recently, Nielsen has provided anecdotal evidence supporting this intuition: In [24] he showed that for the Wheel system over four elements, probing in an order dictated by a dynamically decreasing Banzhaf index gives a better *average* probe complexity than that of a particular fixed strategy. However, proving

that this is a general phenomenon for all quorum systems, either in the worst case or in the average case, is still an open problem.

Acknowledgments. We are grateful to Moni Naor for many stimulating discussions, and in particular for pointing out the connection between our probing strategies and [5]. We thank Jean-Claude Bermond for his help in constructing examples that disproved some early conjectures.

REFERENCES

- [1] D. AGRAWAL AND A. EL-ABBADI, *An efficient and fault-tolerant solution for distributed mutual exclusion*, ACM Trans. Comput. Sys., 9 (1991), pp. 1–20.
- [2] D. BARBARA AND H. GARCIA-MOLINA, *The reliability of vote mechanisms*, IEEE Trans. Comput., 36 (1987), pp. 1197–1208.
- [3] R. A. BAZZI, *Planar quorums*, in Proceedings of the 10th International Workshop on Distributed Algorithms, Bologna, Italy, 1996, Lecture Notes in Comput. Sci. 1151, Springer-Verlag, Berlin, pp. 251–268.
- [4] D. BEAVER AND A. WOOL, *Quorum-based secure multi-party computation*, in Advances in Cryptology—EUROCRYPT’98, Espoo, Finland, 1998, Lecture Notes in Comput. Sci. 1403, K. Nyberg, ed., Springer-Verlag, Berlin, pp. 375–390.
- [5] M. BLUM AND R. IMPAGLIAZZO, *Generic oracles and oracle classes*, in Proceedings of the 28th IEEE Symposium on Foundations of Computer Science, 1987, pp. 118–126.
- [6] S. B. DAVIDSON, H. GARCIA-MOLINA, AND D. SKEEN, *Consistency in partitioned networks*, ACM Comput. Surveys, 17 (1985), pp. 341–370.
- [7] P. ERDŐS AND L. LOVÁSZ, *Problems and results on 3-chromatic hypergraphs and some related questions*, in Infinite and Finite Sets, Proc. Colloq. Math. Soc. János Bolyai 10, North-Holland, Amsterdam, 1975, pp. 609–627.
- [8] A. FU, *Enhancing Concurrency and Availability for Database Systems*, Ph.D. thesis, Simon Fraser University, Burnaby, BC, Canada, 1990.
- [9] H. GARCIA-MOLINA AND D. BARBARA, *How to assign votes in a distributed system*, J. ACM, 32 (1985), pp. 841–860.
- [10] D. K. GIFFORD, *Weighted voting for replicated data*, in Proceedings of the 7th Annual ACM Symposium on Operating Systems Principles, Pacific Grove, 1979, ACM, New York, 1979, pp. 150–159.
- [11] Y. HASSIN AND D. PELEG, *Average probe complexity in quorum systems*, in Proceedings of the 20th ACM Symposium on Principles of Distributed Computing, Newport, RI, 2001, pp. 180–189.
- [12] M. P. HERLIHY, *Replication Methods for Abstract Data Types*, Ph.D. thesis, Massachusetts Institute of Technology, MIT/LCS/TR-319, Cambridge, MA, 1984.
- [13] R. HOLZMAN, Y. MARCUS, AND D. PELEG, *Load balancing in quorum systems*, SIAM J. Discrete Math., 10 (1997), pp. 223–245.
- [14] T. IBARAKI AND T. KAMEDA, *A theory of coteries: Mutual exclusion in distributed systems*, IEEE Trans. Parallel Distrib. Systems, 4 (1993), pp. 779–794.
- [15] J. KAHN, M. SAKS, AND D. STURTEVANT, *A topological approach to evasiveness*, Combinatorica, 4 (1984), pp. 297–306.
- [16] D. E. KNUTH, *The Art of Computer Programming, Vol. 1—Fundamental Algorithms*, Addison-Wesley, Reading, MA, 1968.
- [17] A. KUMAR, *Hierarchical quorum consensus: A new algorithm for managing replicated data*, IEEE Trans. Comput., 40 (1991), pp. 996–1004.
- [18] D. E. LOEB, *The fundamental theorem of voting schemes*, J. Combin. Theory Ser. A, 73 (1996), pp. 120–129.
- [19] L. LOVÁSZ, *Coverings and colorings of hypergraphs*, in Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Boca Raton, FL, 1973, Utilitas Mathematica, Winnipeg, MB, Canada, 1973, pp. 3–12.
- [20] M. MAEKAWA, *A \sqrt{n} algorithm for mutual exclusion in decentralized systems*, ACM Trans. Comput. Sys., 3 (1985), pp. 145–159.

- [21] M. NAOR AND A. WOOL, *Access control and signatures via quorum secret sharing*, IEEE Trans. Parallel Distrib. Systems, 9 (1998), pp. 909–922.
- [22] M. NAOR AND A. WOOL, *The load, capacity and availability of quorum systems*, SIAM J. Comput., 27 (1998), pp. 423–447.
- [23] M. L. NEILSEN, *Quorum Structures in Distributed Systems*, Ph.D. thesis, Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, 1992.
- [24] M. L. NEILSEN, *A dynamic probe strategy for quorum systems*, in Proceedings of the 17th International Conference on Distributed Computing Systems, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 95–99.
- [25] G. OWEN, *Game Theory*, 2nd ed., Academic Press, New York, 1982.
- [26] D. PELEG AND A. WOOL, *The availability of quorum systems*, Inform. and Comput., 123 (1995), pp. 210–223.
- [27] D. PELEG AND A. WOOL, *How to be an efficient snoop, or the probe complexity of quorum systems*, in Proceedings of the 15th ACM Symposium on Principles of Distributed Computing, Philadelphia, PA, 1996, pp. 290–299.
- [28] D. PELEG AND A. WOOL, *Crumbling walls: A class of practical and efficient quorum systems*, Distrib. Comput., 10 (1997), pp. 87–98.
- [29] K. G. RAMAMURTHY, *Coherent Structures and Simple Games*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
- [30] M. RAYNAL, *Algorithms for Mutual Exclusion*, MIT Press, Cambridge, MA, 1986.
- [31] R. L. RIVEST AND J. VUILLEMIN, *On recognizing graph properties from adjacency matrices*, Theoret. Comput. Sci., 3 (1976), pp. 371–384.
- [32] A. L. ROSENBERG, *On the time required to recognize properties of graphs: A problem*, SIGACT News, 5 (1973), pp. 15–16.
- [33] F. B. SCHNEIDER, *What good are models and what models are good?*, in Distributed Systems, S. Mullender, ed., ACM Press, New York, Addison-Wesley, Reading, MA, 1993, pp. 17–26.
- [34] R. H. THOMAS, *A majority consensus approach to concurrency control for multiple copy databases*, ACM Trans. Database Systems, 4 (1979), pp. 180–209.
- [35] A. WOOL, *Quorum Systems for Distributed Control Protocols*, Ph.D. thesis, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel, 1996.
- [36] A. WOOL, *Quorum systems in replicated databases: Science or fiction?*, Bull. IEEE Technical Committee on Data Engineering, 21 (1998), pp. 3–11; also available online from <http://www.research.microsoft.com/research/db/debull/>.

APPROXIMATING BANDWIDTH BY MIXING LAYOUTS OF INTERVAL GRAPHS*

D. KRATSCH[†] AND L. STEWART[‡]

Abstract. We examine the bandwidth problem in circular-arc graphs, chordal graphs with a bounded number of leaves in the clique tree, and k -polygon graphs (fixed k). We show that all of these graph classes admit efficient approximation algorithms which are based on exact or approximate bandwidth layouts of related interval graphs. Specifically, we obtain a bandwidth approximation algorithm for circular-arc graphs that executes in $O(n \log^2 n)$ time and has performance ratio 2, which is the best possible performance ratio of any polynomial time bandwidth approximation algorithm for circular-arc graphs. For chordal graphs with at most k leaves in the clique tree, we obtain a performance ratio of $2k$ in $O(k(n+m))$ time, and our algorithm for k -polygon graphs has performance ratio $2k^2$ and runs in time $O(n^3)$.

Key words. interval graphs, circular-arc graphs, chordal graphs, polygon graphs, bandwidth problem, approximation algorithm

AMS subject classifications. 05C78, 05C85, 68R10, 68W25

PII. S0895480199359624

1. Introduction. A *layout* of a graph $G = (V, E)$ is an assignment of distinct integers from $\{1, \dots, n\}$ to the elements of V . Equivalently, a layout L may be thought of as an ordering $L(1), L(2), \dots, L(n)$ of V , where $|V| = n$. We shall use $<_L$ to denote the ordering of the elements in a layout L . The *width* of a layout L , $b(G, L)$, is the maximum over all edges $\{u, v\}$ of G of $|L(u) - L(v)|$. That is, it is the length of the longest edge in the layout. The *bandwidth* of G , $bw(G)$, is the minimum width over all layouts. A *bandwidth layout* for graph G is a layout satisfying $b(G, L) = bw(G)$.

The problem of finding the bandwidth of a graph has applications in sparse matrix computations. An overview of the bandwidth problem is given in [5]. The minimum bandwidth decision problem (Given a graph $G = (V, E)$ and integer k , is $bw(G) \leq k$?) is known to be NP-complete [27], even for trees having maximum degree 3 [15], caterpillars with hairs of length at most 3 [26], and cobipartite graphs [22]. The problem is polynomially solvable for caterpillars with hairs of length 1 and 2 [2], cographs [18], graphs with few P_4 's [24], and interval graphs [19, 25, 29].

To date there was not much known about the approximation hardness of the bandwidth minimization problem for graphs in general. Recently, Feige presented an approximation algorithm with performance ratio $O(\log^{9/2} n)$ [12]. Very recently, Unger has shown in [30] that, assuming $P \neq NP$, there is no polynomial time approximation algorithm with constant performance ratio for the bandwidth minimization problem for graphs, even when the inputs are restricted to a special class of trees known as caterpillars of hairlength 3.

*Received by the editors July 30, 1999; accepted for publication (in revised form) March 22, 2002; published electronically July 16, 2002. This work appeared in a preliminary form in *Proceedings of the 16th International Symposium on Theoretical Aspects in Computer Science*, Trier, Germany, 1999, and was partially supported by the NSERC.

<http://www.siam.org/journals/sidma/15-4/35962.html>

[†]Fakultät für Mathematik und Informatik, Friedrich-Schiller-Universität, 07740 Jena, Germany. Current address: Université de Metz, Laboratoire d'Informatique Théorique et Appliquée, 57045 Metz Cedex 01, France (kratsch@lita.univ-metz.fr).

[‡]Department of Computing Science, University of Alberta, Edmonton, AB, Canada, T6G 2E8 (stewart@cs.ualberta.ca).

Since the bandwidth minimization problem remains NP-complete for such simple classes of graphs, and since no polynomial time algorithm for approximating the bandwidth of general graphs, or even trees, to within a constant factor exists unless $P=NP$, it is worthwhile to investigate approximation algorithms for this problem on restricted classes of graphs. Some results in this direction have been presented in [22].

In this paper, we examine the bandwidth problem in circular-arc graphs, chordal graphs with a bounded number of leaves in the clique tree, and k -polygon graphs (fixed k). All of these graph classes admit efficient approximation algorithms which are based on exact or approximate bandwidth layouts of related interval graphs.

Specifically, we obtain a bandwidth approximation algorithm for circular-arc graphs that has performance ratio 2 and executes in $O(n \log^2 n)$ time or performance ratio 4 while taking $O(n)$ time. For chordal graphs with at most k leaves in the clique tree, we obtain a performance ratio of $2k$ in $O(k(n+m))$ time, and our algorithm for k -polygon graphs has performance ratio $2k^2$ and runs in time $O(n^3)$.

Finally, it is worth mentioning that our approximation algorithm with performance ratio 2 for circular-arc graphs has *optimal* performance ratio, since there is no polynomial time bandwidth approximation algorithm for circular-arc graphs with performance ratio $2 - \epsilon$ for any $\epsilon > 0$ unless $P=NP$ [30].

2. Preliminaries. For $G = (V, E)$, we will denote $|V|$ as n and $|E|$ as m . We sometimes refer to the vertex set of G as $V(G)$ and the edge set as $E(G)$. We let $N(v)$ denote the set of vertices adjacent to v . The *degree* of a vertex v , $degree(v)$, is the number of vertices adjacent to v . $\Delta(G)$ denotes the maximum degree of a vertex in graph G . The subgraph of $G = (V, E)$ induced by $V' \subseteq V$ will be referred to as $G[V']$.

The following well-known lower bound on the bandwidth of a graph is in [6].

LEMMA 1 (the degree bound [7]). *For any graph G , $bw(G) \geq \Delta(G)/2$.*

The distance in graph $G = (V, E)$ between two vertices $u, v \in V$, $d_G(u, v)$, is the length of a shortest path between u and v in G . For any graph $G = (V, E)$, the d th power of G , G^d , is the graph with vertex set V and edge set $\{\{u, v\} | d_G(u, v) \leq d\}$.

LEMMA 2 (the distance bound [22], also attributed in part to [7] in [5]). *Let G and H be graphs with the same vertex set V , such that $E(G) \subseteq E(H) \subseteq E(G^d)$ or $E(H) \subseteq E(G) \subseteq E(H^d)$ for an integer $d \geq 1$, and let L be an optimal layout for H , i.e., $b(H, L) = bw(H)$. Then L approximates the bandwidth of G by a factor of d , i.e., $b(G, L) \leq d \cdot bw(G)$.*

Many references, including [17], contain comprehensive overviews of the many known structural and algorithmic properties of interval graphs.

DEFINITION. *A graph $G = (V, E)$ is an interval graph if there is a one-to-one correspondence between V and a set of intervals of the real line such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the intervals corresponding to u and v have a nonempty intersection.*

A set of intervals whose intersection graph is G is termed an *interval model* for G . Many algorithms exist which, given a graph $G = (V, E)$, determine whether or not G is an interval graph and, if so, construct an interval model for it in $O(n+m)$ time (see, for example, [4, 8]). We assume that an interval model is given by a left endpoint and a right endpoint for each interval, namely, $left(v)$ and $right(v)$ for all $v \in V$. Furthermore, we assume that we are also given a sorted list of the endpoints and that the endpoints are distinct. We will sometimes blur the distinction between an interval and its corresponding vertex, when no confusion can arise.

Polynomial time algorithms for computing the exact bandwidth of an interval

graph have been given in [19, 25, 29]. For an interval graph with n vertices, Kleitman and Vohra’s algorithm solves the decision problem ($bw(G) \leq k?$) in $O(nk)$ time and can be used to produce a bandwidth layout in $O(n^2 \log n)$ time, and Sprague has shown how to implement Kleitman and Vohra’s algorithm to answer the decision problem in $O(n \log n)$ time and thus produce a bandwidth layout in $O(n \log^2 n)$ time.

The following two lemmas demonstrate that, for interval graph G , a layout L with $b(G, L) \leq 2 \cdot bw(G)$ can be obtained in time $O(n)$, assuming the sorted interval endpoints are given. The second proof is similar to the first and is therefore omitted.

LEMMA 3. *Given an interval graph G , the layout L consisting of vertices ordered by right endpoints of corresponding intervals has $b(G, L) \leq 2 \cdot bw(G)$.*

Proof. Let L be the layout of vertices ordered by right interval endpoints. We first observe that, for all $u, v \in V$ such that $\{u, v\} \in E$ and $u <_L v$, all vertices between u and v in L are adjacent to v . Now consider a longest edge in L , i.e., an edge $\{u, v\}$ such that $|L(u) - L(v)| = b(G, L)$. Assume, without loss of generality, that $u <_L v$. From the previous observation, it must be that $degree(v) \geq L(v) - L(u) = b(G, L)$. Now the degree bound (Lemma 1) implies $bw(G) \geq b(G, L)/2$. \square

LEMMA 4. *Given an interval graph G , the layout L consisting of vertices ordered by left endpoints of corresponding intervals has $b(G, L) \leq 2 \cdot bw(G)$.*

We will use the following lemma in subsequent sections of the paper.

LEMMA 5. *Let I be a set of intervals on the real line corresponding to interval graph $G = (V, E)$. Let p_1 be a point on the line such that at least one interval endpoint is to the left of p_1 and only left endpoints are to the left of p_1 . Let p_2 be a point on the line such that at least one interval endpoint is to the right of p_2 and only right endpoints are to the right of p_2 . Let C_1 be the set of all intervals that contain p_1 , and let C_2 be the set of all intervals that contain p_2 . If L is a layout for G in which vertices are ordered by increasing left endpoints of corresponding intervals or by increasing right endpoints, or if L is a layout produced by Kleitman and Vohra’s bandwidth algorithm [19], then*

- (i) for all $v \in C_1$: $\{v, L(1)\} \in E$, and
- (ii) for all $v \in C_2$: $\{v, L(n)\} \in E$.

Proof. Part (i) for the left endpoint ordering follows from the fact that $L(1) \in C_1$ and C_1 is a clique. In the other two layouts, $L(1)$ is the interval with the smallest right endpoint. This interval is either in C_1 or is contained in all intervals of C_1 . Thus, (i) holds for the three layouts.

Part (ii) follows immediately for the right endpoint layout, since $L(n) \in C_2$. In the left endpoint order, $L(n)$ is either in C_2 or contained in all intervals of C_2 , implying (ii).

Finally, we prove (ii) for Kleitman–Vohra layouts. Please refer to the algorithm of [19]. Consider the moment when the vertex of C_2 with largest left endpoint, c , is labelled. If only vertices of C_2 remain to be labelled, then the last vertex will be an element of C_2 , and we are done. Otherwise, there is an interval i with a smaller right endpoint that remains to be labelled. This implies that $c \in S_{j_0}^q$ was chosen in Step 8, and $i \notin S_{j_0}^q$. Since $i \notin S_{j_0}^q$, we have $q + j_0 < n$. Thus, $M(c) < n$, and there is some vertex already labelled that is adjacent to c but not to i ; otherwise, we contradict the current choice of c . Thus, the interval i is properly contained in c and, therefore, i is properly contained in all intervals corresponding to vertices of C_2 . This completes the proof. \square

3. Circular-arc graphs. Circular-arc graphs are the intersection graphs of arcs on a circle. Thus, a graph $G = (V, E)$ is a circular-arc graph if and only if it has a (not

necessarily unique) circular-arc model or representation, consisting of a set of arcs on a circle, such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the arcs corresponding to u and v have a nonempty intersection. In such a model, we assume, without loss of generality, that the arc endpoints are distinct, and we label the endpoints from 1 to $2n$ in clockwise order around the circle, starting at an arbitrary endpoint. Thus, each vertex $v \in V$ corresponds to an arc given by its counterclockwise endpoint, $ccw(v)$, and its clockwise endpoint, $cw(v)$. We refer to any segment of the circle by its two endpoints and the direction of traversal; i.e., $[p_1, p_2]_{cw}$ refers to the closed arc covered by a clockwise traversal beginning at p_1 and ending at p_2 . The arc $[p_1, p_2]_{ccw}$ is the set of all points in a counterclockwise traversal from p_1 to p_2 , and parentheses will indicate that the arc is open at one or both ends. Note that, for any two points on the circle, p_1 and p_2 , the arcs $[p_1, p_2]_{cw}$ and $[p_1, p_2]_{ccw}$ cover the entire circle, and their intersection is $\{p_1, p_2\}$.

Eschen and Spinrad [11] have given an $O(n^2)$ algorithm which determines whether or not an n -vertex graph is a circular-arc graph. If so, the algorithm produces a circular-arc model for the graph. Our algorithms assume that the input circular-arc graph is given as a set of arcs on a circle. We are not aware of any previous results on the bandwidth of circular-arc graphs.

Henceforth, we will refer to a set of $2n$ scanpoints on the circle, none of which is an arc endpoint, such that exactly one of these points is between each consecutive pair of arc endpoints. We shall label these points from 1 to $2n$ in clockwise order, beginning at any one.

Our bandwidth approximation algorithm works as follows for a circular-arc graph G . Roughly speaking, we cut the circular-arc representation in half to form two equal-sized interval graphs, compute exact or approximate bandwidth layouts for the two interval graphs, and then mix the two layouts to form an approximate bandwidth layout for G .

Let $G = (V, E)$ be a circular-arc graph with corresponding circular-arc representation. The first step is to find a scanpoint p on the circle such that $|C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|$, where C_1 is the set of arcs that contain scanpoint 1, C_2 is the set of arcs that contain scanpoint p , A is the set of arcs entirely contained in $(1, p)_{cw}$, and B is the set of arcs entirely contained in $(1, p)_{ccw}$. Note that $C_1 \cup C_2 \cup A \cup B = V$. We will use scanpoints 1 and p to cut the circle and create two equal-sized interval graphs.

ALGORITHM 1. **Procedure** *FIND* p .

Let $C_1 \leftarrow C_2 \leftarrow$ all arcs that contain scanpoint 1; $A \leftarrow \emptyset$; $B \leftarrow V \setminus C_1$

$a \leftarrow |C_1|$; $b \leftarrow n$ { $a = |C_1 \cup C_2 \cup A|$; $b = |C_1 \cup C_2 \cup B|$ }

$p \leftarrow 1$

repeat until $a = b$ **or** $p = 2n$

{ **Invariant:** $a \leq b$ }

{ **Variant:** $2n - p$ }

$p \leftarrow p + 1$

if the endpoint between $p - 1$ and p is a *ccw* endpoint (say of arc i) **then**

$C_2 \leftarrow C_2 \cup \{i\}$

if $i \notin C_1$ **then**

$B \leftarrow B \setminus \{i\}$; $a \leftarrow a + 1$

if between $p - 1$ and p is a *cw* endpoint (of arc i) **then**

$C_2 \leftarrow C_2 \setminus \{i\}$

if $i \notin C_1$ **then**

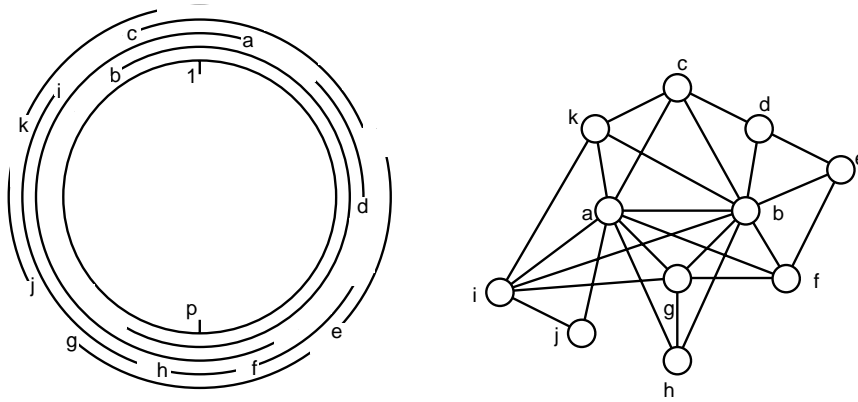


FIG. 1. A set of arcs on a circle and the corresponding circular-arc graph.

$A \leftarrow A \cup \{i\}; \quad b \leftarrow b - 1$
 { Now C_2 is the set of arcs that contain point p }
 { $|C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|$ }.

Claim 1. Procedure FIND p will terminate with $a = b$.

Proof. We leave it to the reader to verify the stated invariant and variant. If the loop terminates with $p = 2n$, then all arc endpoints will have been examined. For all arcs, except those of C_1 , a will have been incremented by 1 and b will have been decremented by 1. Let a_i and a_f be the initial and final values, respectively, of variable a and b_i and b_f the initial and final values, respectively, of variable b . Upon termination of the loop with $p = 2n$, $a_f = a_i + n - |C_1| = |C_1| + n - |C_1| = n$ and $b_f = b_i - (n - |C_1|) = n - n + |C_1| = |C_1|$. However, then $b_f < a_f$ (assuming $C_1 \neq V$), contradicting our invariant. \square

We may assume that A and B will be nonempty; otherwise, G can be partitioned into two cliques, one of which must have size at least $n/2$, implying (by Lemma 1) $bw(G) \geq n/2 - 1$. Thus, any layout in which the first and last vertices are not adjacent is a 2-approximation.

A set of arcs on a circle and the corresponding graph are shown in Figure 1, along with possible choices of scanpoints 1 and p . In this example, $C_1 = \{a, b, c\}$, $C_2 = \{a, b, g, h\}$, $A = \{d, e, f\}$, and $B = \{i, j, k\}$.

We now describe how to construct two interval subgraphs of G by cutting the circle at scanpoints 1 and p . We wish to cut the circle and the arcs of C_1 and C_2 at scanpoints 1 and p , producing two line segments, each with a set of intervals that correspond to an interval graph. However, if any arc, say v , contains both scanpoints 1 and p , then it covers one entire part of the circle (i.e., $[1, p]_{cw}$ or $[1, p]_{ccw}$) and appears as two disconnected pieces in the other part. Thus, this second part of the circle may not correspond to an interval subgraph, as vertex v is represented by two disconnected intervals. We eliminate this problem by shrinking v 's arc on the circle so that it no longer contains p and thus v is removed from C_2 . The altered set of arcs might not represent all of the edges of G ; specifically, some edges between v and elements of A (or B) may be missing. Let E' denote edges of G that are not represented by the changed arcs. Note that the sets $C_1 \cup C_2 \cup A$ and $C_1 \cup C_2 \cup B$ remain unchanged. These alterations, applied to the circular-arc model of Figure 1, yield the set of arcs shown in Figure 2. After the alterations, C_2 is changed to $\{g, h\}$,

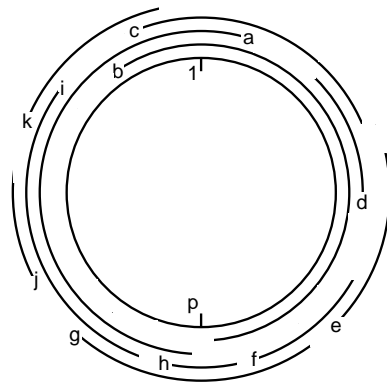


FIG. 2. Altering the circular-arc model.

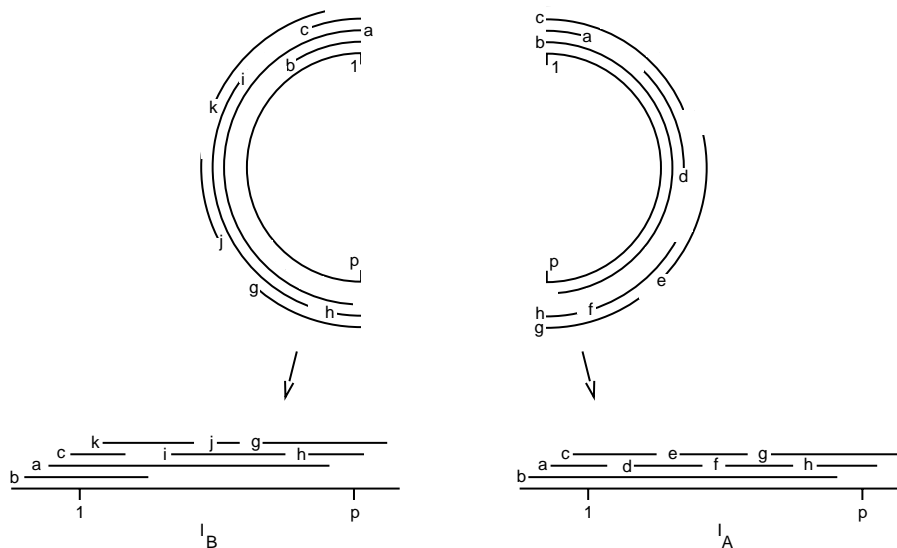


FIG. 3. Cutting the circular-arc model to form two interval graphs.

C_1 , A , and B remain unchanged, and $E' = \{\{a, f\}, \{b, i\}\}$.

Now we can cut the circle and the arcs of C_1 and C_2 at scanpoints 1 and p , producing two line segments, $[1, p]_{cw}$ and $[1, p]_{ccw}$. The arcs of the circular-arc model become intervals on the two lines. Let I_A (respectively, I_B) be the resulting set of intervals on the line segment $[1, p]_{cw}$ (respectively, $[1, p]_{ccw}$). We may assume that the intervals of $C_1 \cup C_2$ are altered slightly in I_A and in I_B without changing intersections, so that interval endpoints are distinct.

Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be the intersection graphs of I_A and I_B , respectively. Now G_A and G_B are both interval graphs and (not necessarily induced) subgraphs of G . Furthermore, $|V_A| = |V_B|$, and $E_A \cup E_B \cup E' = E$. Figure 3 illustrates this process for the example of Figures 1 and 2.

Our method for obtaining an approximate bandwidth layout for a circular-arc graph is to first compute exact or approximate bandwidth layouts, L_A and L_B , for G_A and G_B , respectively, and then mix the two layouts.

Different methods of computing L_A and L_B yield different approximation bounds and time complexities for our algorithm.

Regardless of how we obtain L_A and L_B , the mixing is done as follows. Let $k = |C_1 \cup C_2 \cup A| = |C_1 \cup C_2 \cup B|$. Given

$$L_A = L_A(1), L_A(2), \dots, L_A(k)$$

and

$$L_B = L_B(1), L_B(2), \dots, L_B(k)$$

we begin by producing

$$L_M = L_A(1), L_B(1), L_A(2), L_B(2), \dots, L_A(k), L_B(k).$$

For convenience, we will refer to elements of L_A as having the color *red* and elements of L_B as having the color *blue*. Notice that L_M will contain two copies of each vertex of $C_1 \cup C_2$ —one red and one blue. For each $v \in C_1 \cup C_2$, we shall distinguish between the two copies of v in L_M as follows: the red copy will be referred to as v_{red} and the blue copy as v_{blue} . Each vertex of $A \cup B$ occurs only once in L_M .

From L_M , we produce L by deleting the leftmost copy of each vertex of C_1 and the rightmost copy of each vertex of C_2 . Recall that we constructed C_1 and C_2 so that no vertex appears in both. Thus, L is a layout for G . We now prove a bound on the width of L in terms of the widths of L_A and L_B .

LEMMA 6. *Let $G = (V, E)$ be a circular-arc graph, and let $I_A, I_B, G_A,$ and G_B be constructed, as previously described, from a circular-arc model for G . Let L_A and L_B be layouts for G_A and G_B , respectively, satisfying*

- for all $v \in C_1$: $\{v, L_A(1)\}, \{v, L_B(1)\} \in E$, and
- for all $v \in C_2$: $\{v, L_A(k)\}, \{v, L_B(k)\} \in E$.

Let L_M and L be obtained from L_A and L_B as previously described. Then

$$b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)].$$

Proof. We will consider an arbitrary edge of G , $\{u, v\} \in E$. We first observe that if u and v have the same color, say red, then $|L(u) - L(v)| \leq |L_M(u) - L_M(v)| = 2 \cdot |L_A(u) - L_A(v)|$. Such edges, therefore, cannot contradict the claim. We shall refer to such edges as red edges or blue edges, depending upon the color of the endpoints. Similarly, any edge for which we can find a longer red or blue edge in L_M cannot contradict the claim.

Consider the edge $\{u, v\} \in E$, where $u <_L v$. We must show that $|L(u) - L(v)| \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)]$.

Case 1. The intervals corresponding to u and v intersect in I_A or I_B or both.

Hence $\{u, v\} \in E \setminus E'$. Suppose, without loss of generality, that the intervals intersect in I_A . If u and v are both red, then our earlier observation applies, and we are done.

Next, suppose that u is red and v is blue. When L was formed from L_M , v_{red} must have been deleted. If v_{red} is to the right of v_{blue} in L_M , then there is a longer red edge $\{u, v\}$ in L_M , and this completes the proof. Suppose v_{red} is to the left of v_{blue} in L_M . This implies that $v \in C_1$, since the leftmost copy was deleted from L_M to form L . However, then v_{blue} is adjacent to the first blue vertex in L_M , implying that $|L(v) - L(u)| \leq |L_M(v_{blue}) - L_M(2)| \leq 2 \cdot |L_B(v) - L_B(1)|$.

Now consider the case where u is blue and v is red. The red copy of u has been deleted. If u_{red} is to the left of u_{blue} in L_M , then there is a longer red edge in L_M . Otherwise, we have $u \in C_2$. However, then u_{blue} is adjacent to the last vertex of L_M , giving a longer blue edge.

Finally, we consider the case where u and v are both blue. If the corresponding intervals intersect in I_B , then we are done by the previous argument. Otherwise, one of u and v is in C_1 and the other is in C_2 . If $u \in C_1$ and $v \in C_2$, then the red edge $\{u_{red}, v_{red}\}$ is longer in L_M than $\{u, v\}$ in L . If $u \in C_2$ and $v \in C_1$, then u_{blue} is adjacent to the last vertex of L_M , giving a blue edge in L_M longer than $\{u, v\}$ in L .

Case 2. The intervals corresponding to u and v intersect neither in I_A nor in I_B .

Hence $\{u, v\} \in E'$. Then it must be that exactly one of the vertices corresponds to an arc which, in the original circular-arc representation, covers all of one side of the circle and extends into the other side covering both scanpoints 1 and p . Assume, without loss of generality, that the arc covers $[1, p]_{ccw}$ and appears as two disconnected arcs in $[1, p]_{ccw}$. In constructing I_B , the part of the arc that covered p and extended into $[1, p]_{ccw}$ was removed. This must be the area where the arcs corresponding to u and v intersected in the original circular-arc representation. This implies that the other arc is in B , and it therefore occurs as a blue vertex only in L_M and in L .

Suppose that u is the arc that was altered. Then $u \in C_1$ and $v \in B$. Thus, it is the rightmost copy of u that remains in L . The red copy of u in L_M is adjacent to all other red vertices, including $L_M(2k-1)$. Thus, if u in L is red, then there is a red edge in L_M that is longer than the $\{u, v\}$ edge in L . If u in L is blue, then u_{red} has a longer edge in L_M to $L_M(2k-1)$.

Now consider the case where v was altered. Then $v \in C_1$ and $u \in B$. The rightmost copy of v from L_M remains in L , and the red copy of v is adjacent to all other red vertices in L_M , including $L_M(1)$. If v in L is red, then the red edge $\{v, L_M(1)\}$ is longer than the edge $\{u, v\}$ in L . If v is blue in L , then v is adjacent to $L_M(2)$ by Lemma 5 and the construction of L_M ; thus, there is a longer blue edge. \square

THEOREM 7. *The bandwidth of a circular-arc graph can be approximated to within a factor of four in $O(n)$ time and to within a factor of two in $O(n \log^2 n)$ time.*

Proof. We have three approximation algorithms for approximating the bandwidth of a circular-arc graph, namely, the algorithm previously described in which

- (i) L_A and L_B are layouts of vertices ordered by left endpoints of intervals,
- (ii) L_A and L_B are layouts of vertices ordered by right endpoints of intervals, or
- (iii) L_A and L_B are layouts computed by Kleitman and Vohra's algorithm.

Algorithms (i) and (ii) have time complexity $O(n)$, provided the sorted arc endpoints are given, and they output a layout L that satisfies $b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)] \leq 4 \cdot bw(G)$.

Algorithm (iii) requires $O(n \log^2 n)$ time but produces a layout L satisfying $b(G, L) \leq 2 \cdot \max[b(G_A, L_A), b(G_B, L_B)] \leq 2 \cdot bw(G)$.

These performance ratios follow from Lemmas 5 and 6 and the fact that any subgraph of graph G has bandwidth not larger than $bw(G)$. \square

4. Chordal graphs with clique trees having a bounded number of leaves.

A graph G is a *chordal* graph if every cycle of length greater than three has a chord. Chordal graphs are exactly the intersection graphs of subtrees in a tree [16]. More precisely, for each chordal graph $G = (V, E)$, there exists a tree T such that

- the vertices of T correspond to the maximal cliques of G , and

- the vertices of T corresponding to cliques of G containing any fixed vertex $v \in V$ induce a subtree T_v of T .

Note the consequence that two vertices of G are adjacent if and only if their corresponding subtrees have nonempty intersection. For a given chordal graph $G = (V, E)$, such a tree, called a *clique tree* for G , will have at most n nodes and can be constructed in $O(n + m)$ time [3].

We use the idea of mixing layouts of interval graphs, as in the previous section. While a circular-arc graph roughly consists of two interval graphs arranged in a circle, a chordal graph may be thought of as several interval graphs arranged in a tree-like structure. We restrict our attention to chordal graphs having a bounded number of leaves in their clique trees. A chordal graph with k leaves in its clique tree may be viewed as a collection of k interval graphs. For a chordal graph $G = (V, E)$ with at most k leaves in the corresponding clique tree, we compute a layout L such that $b(G, L) \leq 2k \cdot bw(G)$.

The method is as follows, assuming a clique tree T has been computed for a given chordal graph $G = (V, E)$.

1. Root T at an arbitrary vertex, r .
2. Let k be the number of leaves of T (excluding r). For each root-to-leaf path P_i in T , the collection of subtrees T_v (for $v \in V$), restricted to P_i , form a set of intervals. Let I_i be this set of intervals in which the left endpoint of each interval is taken to be the one closer to r . Let $G_i = (V_i, E_i)$ be the corresponding interval graph.
3. **for** $i \leftarrow 1$ **to** k **do**
 $L_i \leftarrow$ layout for G_i consisting of V_i ordered by increasing left endpoints of intervals (with ties broken arbitrarily but the same way in all the L_i 's)
4. Mix the L_i 's to form L_M , as follows:
 $L_M \leftarrow L_1(1)L_2(1)L_3(1) \dots L_k(1)L_1(2)L_2(2) \dots L_k(2) \dots$
5. For each vertex $v \in V$ that appears in more than one of the G_i 's, delete all but the rightmost copy of v from L_M . The result is a layout L for G .

The following lemmas apply in the context of the previously described method.

LEMMA 8. *Each G_i is an interval graph.*

Proof. This follows from the construction of the G_i 's and properties of the clique tree. \square

LEMMA 9. $E_1 \cup E_2 \cup \dots \cup E_k = E$.

Proof. If $\{u, v\} \in E$, then u and v occur together in some clique corresponding to a vertex of T . Thus the edge $\{u, v\}$ will occur in every G_i whose corresponding path P_i contains that vertex of T . \square

LEMMA 10. *For all $\{u, v\} \in E$, either*

- *for all $1 \leq i \leq k : u \in V_i$ implies $(v \in V_i \text{ and } \{u, v\} \in E_i)$, or*
- *for all $1 \leq i \leq k : v \in V_i$ implies $(u \in V_i \text{ and } \{u, v\} \in E_i)$.*

Proof. Let $\{u, v\} \in E$. Then T_u and T_v intersect. Let c_{uv} be the vertex of T , closest to r , at which T_u and T_v intersect. c_{uv} is the closest to r vertex for at least one of T_u and T_v ; otherwise, we contradict our choice of c_{uv} , since the path from c_{uv} to r in T is unique and since T_u and T_v are both connected.

Suppose c_{uv} is the vertex of T_u closest to r in T . Then, for any V_i that contains u , the corresponding path P_i must contain c_{uv} , and the conclusion follows.

Similarly, if c_{uv} is the vertex of T_v closest to r , then, for every V_i containing v , the corresponding path P_i contains c_{uv} . \square

LEMMA 11. *Each G_i is an induced subgraph of G .*

Proof. This follows by an argument similar to the previous proof. \square

LEMMA 12. $b(G, L) \leq 2k \cdot bw(G)$.

Proof. Let $\{u, v\} \in E$ and consider the length of $\{u, v\}$ in L , i.e., $|L(u) - L(v)|$. Assume, without loss of generality, that $u <_L v$. If the copies of u and v remaining in L are from the same interval subgraph G_i , then

$$\begin{aligned} |L(u) - L(v)| &\leq |L_M(u) - L_M(v)| \\ &\leq k \cdot |L_i(u) - L_i(v)| \\ &\leq 2k \cdot bw(G_i) \\ &\leq 2k \cdot bw(G). \end{aligned}$$

Suppose the occurrences of u and v in L are from different interval subgraphs, G_u and G_v , respectively. Since $\{u, v\} \in E$, we know by Lemma 10 that

- $v \in G_u$ and $\{u, v\} \in G_u$, or
- $u \in G_v$ and $\{u, v\} \in G_v$.

If $u \in G_v$ then the occurrence of u in G_v is to the left (in L_M) of the occurrence of u in G_u . Thus

$$\begin{aligned} |L(u) - L(v)| &\leq |L_M(u \text{ of } G_v) - L_M(v \text{ of } G_v)| \\ &\leq k \cdot |L_v(u) - L_v(v)| \\ &\leq 2k \cdot bw(G_v) \\ &\leq 2k \cdot bw(G). \end{aligned}$$

Otherwise, $u \notin G_v$ and $v \in G_u$, implying that the vertex of T_v closest to r is closer to r than the vertex of T_u closest to r . That is, in I_u , $left(v) < left(u)$, and hence $v <_{L_u} u$.

Let f_{uv} be the last vertex of T (i.e., farthest from the root) that is in both G_u and G_v . The set of left endpoints from r to f_{uv} are identical in both I_u and I_v . Suppose there are q of them. Then, in L_M , both occurrences of v appear in the first $k \cdot q$ positions, and the occurrence of u from G_u is to the right. This contradicts that the occurrences of u and v under consideration satisfy $u <_L v$. \square

THEOREM 13. *Let $G = (V, E)$ be a chordal graph having a clique tree with at most k leaves. Then a layout L for G satisfying $b(G, L) \leq 2k \cdot bw(G)$ can be computed in $O(k(n + m))$ time.*

Proof. The proof follows from the previous discussion. \square

Recently, Fomin [13] showed how to improve the previous bound by choosing the root r of the clique tree T such that at most $\frac{2}{3}k$ of the resulting interval graphs can be laid out on either side of the vertices of r . The layout L obtained after mixing the L_i 's and removing duplicate vertices has $b(G, L) \leq \frac{4}{3}k \cdot bw(G)$.

Since the bandwidth problem remains NP-complete for trees, a subclass of chordal graphs, it is worth mentioning that our algorithm outputs a layout L satisfying $b(G, L) \leq k$ if G is a tree with at most k leaves. Notice that Ando, Kaneko, and Gervacio showed in [1] that every tree with k leaves has bandwidth at most $\lceil k/2 \rceil$. Furthermore, their construction can easily be transformed into an efficient algorithm to compute a layout of width at most $\lceil k/2 \rceil$.

5. k -polygon graphs for fixed k . A *triangulation* of a graph G is a chordal graph H with the same vertex set as G such that G is a subgraph of H . A triangulation

H of a graph G is called a *minimal triangulation* of G if no proper subgraph of H is a triangulation of G .

In this section we combine results of the previous section with results on minimal triangulations of k -polygon graphs as follows. First, we generalize a result on minimal triangulations of AT-free graphs in [22] to show that every minimal triangulation H of a k -polygon graph G is a spanning subgraph of G^k (Theorem 19). Second, we use a representation theorem for minimal triangulations of a circle graph provided in [23] to obtain a representation theorem for minimal triangulations of k -polygon graphs (Theorem 22). Then we show how to transform any k -polygon graph G into a minimal triangulation H of G (and thus H is a chordal graph) such that H has a clique tree with at most k leaves. Combining all this with Lemma 2, and the approximation algorithm of the previous section, we obtain an $O(n^3)$ approximation algorithm for the bandwidth of k -polygon graphs which has performance ratio $2k^2$ (or $\frac{4}{3}k^2$, in light of Fomin's improvement).

A graph $G = (V, E)$ is a k -polygon graph if it is the intersection graph of chords inside a convex k -polygon, where each chord has its endpoints on two different sides of the polygon. A k -polygon representation, or diagram, for $G = (V, E)$ is a k -sided convex polygon together with a set of chords such that, for all $u, v \in V$, $\{u, v\} \in E$ if and only if the chords corresponding to u and v cross.

Circle graphs are the intersection graphs of chords inside a circle. A circle model, or diagram, for circle graph $G = (V, E)$ is a set of chords in a circle such that two vertices are adjacent in G if and only if their corresponding chords cross. Clearly, every polygon representation of a graph G can also be seen as a circle model of G . Thus, for each $k \geq 2$, every k -polygon graph is a circle graph. (Permutation graphs are to be considered as 2-polygon graphs.)

There is an $O(n^2)$ algorithm [28] which determines whether or not a given graph is a circle graph and, if so, produces a circle model for it. Given a graph $G = (V, E)$, it can be determined in $O(|V|^k)$ time whether or not G is a k -polygon graph and, if so, a polygon representation can be constructed [10]. However, given a circle graph G , the problem of determining the minimum k such that G is a k -polygon graph remains NP-complete [10].

Our algorithm assumes that a k -polygon representation for the input graph is provided.

All of the notation in this section is either identical to that of [22] and [23] or inspired by those two papers. (See also [20].)

Let $G = (V, E)$ be a graph and a, b two nonadjacent vertices of G . The set $S \subseteq V$ is an a, b -separator if the removal of S separates a and b in distinct connected components. If no proper subset of S is an a, b -separator, then S is a *minimal a, b -separator*. A *minimal separator* is a set of vertices S that is a minimal a, b -separator.

LEMMA 14 (see [9]). *Let S be a minimal a, b -separator of the graph $G = (V, E)$, and let C_a and C_b be the connected components of $G[V \setminus S]$ containing a and b , respectively. Then every vertex of S has at least one neighbor in C_a and at least one neighbor in C_b .*

We denote by $\mathfrak{Sep}(H)$ the set of all minimal separators of a graph H . We shall need the following properties of minimal triangulations of a graph.

THEOREM 15 (see [22]). *A triangulation H of a graph G is a minimal triangulation of G if and only if the following three conditions are satisfied:*

1. *If a and b are nonadjacent vertices of H , then every minimal a, b -separator of H is also a minimal a, b -separator of G .*

2. If S is a minimal separator of H and C a connected component of $H[V \setminus S]$, then the vertex set of C induces a connected component in $G[V \setminus S]$.
3. $H = G_{\mathfrak{Sp}(H)}$, where $G_{\mathfrak{Sp}(H)}$ is the graph obtained from G by adding edges between every pair of vertices contained in the same set S for any $S \in \mathfrak{Sp}(H)$.

To obtain an algorithm to approximate the bandwidth of k -polygon graphs, in a first step we generalize definitions and results of [22] to show that every minimal triangulation H of a k -polygon graph G is a subgraph of G^k .

DEFINITION. A minimal separator S is d -good if, for every nonadjacent pair x and y in S , $d_G(x, y) \leq d$. A triangulation H of G is d -good if, for every edge $\{a, b\}$ in H , $d_G(a, b) \leq d$; i.e., H is a subgraph of G^d .

The following theorem is a consequence of the characterization of minimal triangulations given in Theorem 15.

THEOREM 16. If every minimal separator of a graph G is d -good, then every minimal triangulation H of G is d -good.

Proof. Let $\{a, b\}$ be an edge of H but not an edge of G . By Theorem 15, $H = G_{\mathfrak{Sp}(H)}$. Hence there is a minimal separator S of H such that $\{a, b\} \subseteq S$. By Theorem 15, S is also a minimal separator of G . Therefore S is d -good and $d_G(a, b) \leq d$.

Consequently, H is d -good. \square

LEMMA 17. Let G be a graph without a chordless cycle of length greater than $2k + 1$. Then every minimal separator of G is k -good.

Proof. Assume there is some minimal separator S containing nonadjacent vertices x and y such that $d_G(x, y) > k$. Now, by Lemma 14, we can find an x, y -path in C_x and one in C_y . If we choose shortest such paths, then their union is a chordless cycle of length at least $2(k + 1)$, a contradiction. \square

LEMMA 18. Let G be a k -polygon graph. Then G has no chordless cycle of length greater than $2k$.

Proof. It is proved in [14] that chordless cycles have unique representations as chords in a circle. Suppose G has a chordless cycle of length at least $2k + 1$ and consider the unique representation as chords in a circle. The number of chord endpoints must be at least $2(2k + 1) = 4k + 2$. Each side of the k -polygon can contain at most four chord endpoints; otherwise, the two endpoints of a chord would have to be on the same side. Thus there must be at least $\lceil \frac{4k+2}{4} \rceil = k + 1$ sides. \square

THEOREM 19. Every minimal triangulation H of a k -polygon graph G is k -good, and thus H is a subgraph of G^k .

Proof. By Lemma 17 and 18, every minimal separator of a k -polygon graph G is k -good. Thus, by Theorem 16, every minimal triangulation of G is k -good. \square

In a second step we use a representation theorem for the minimal triangulations of a circle graph given in [23] to obtain a similar theorem for k -polygon graphs. We shall need some preparations.

Assume that an n -vertex circle graph is given as a set of chords in a circle. Between each two consecutive endpoints of chords, add a point called a *scanpoint*. Let Z be the set of $2n$ scanpoints. A *scanline* is a chord of the circle connecting two scanpoints. Let c_1 and c_2 be two chords of the circle model. A scanline s is *between* c_1 and c_2 if every path from an endpoint of c_1 to an endpoint of c_2 along the circle passes through a scanpoint of s . For any scanline s , we denote by $S(s)$ the set of all vertices v of G for which the corresponding chord intersects s .

THEOREM 20 (see [21]). Let a and b be nonadjacent vertices of the circle graph $G = (V, E)$. For every minimal a, b -separator S of G , there exists a scanline s between

the chords of a and b such that $S = S(s)$.

Note that this implies that, for every minimal a, b -separator S of a k -polygon graph G , there is a scanline s with $S = S(s)$ such that the endpoints of s are on two different sides of the polygon.

In [23] Kloks, Kratsch, and Wong give the following representation theorem for all minimal triangulations of a circle graph in terms of planar triangulations of the polygon $\mathcal{P}(Z)$, which is the convex polygon with vertex set Z .

THEOREM 21 (see [23]). *Let $G = (V, E)$ be a circle graph given as a set of chords in a circle, and let Z be the corresponding set of scanpoints. Then for every minimal triangulation H of G there is a planar triangulation T of the polygon $\mathcal{P}(Z)$ such that $H = H(T)$, where $H(T)$ is the graph with vertex set V , and vertices u and v are adjacent in $H(T)$ if there exists a triangle in T that is intersected by the chords corresponding to u and v .*

Let $G = (V, E)$ be a k -polygon graph and thus a circle graph. Consider a k -polygon representation of G consisting of a set of chords C inside a k -sided polygon \mathcal{P}_G . Let Z be the set of scanpoints on \mathcal{P}_G , and let $\mathcal{P}(Z)$ be the convex polygon with vertex set Z .

THEOREM 22. *Let $G = (V, E)$ be a k -polygon graph given as a set of chords in a k -polygon, and let Z be the corresponding set of scanpoints. Then for every minimal triangulation H of G there is a planar triangulation T of the polygon $\mathcal{P}(Z)$ such that*

- every diagonal in T has endpoints on two different sides of the k -polygon, and
- $H = H(T)$, where $H(T)$ is the graph with vertex set V , and vertices u and v are adjacent in $H(T)$ if there exists a triangle Q in T that is intersected by the chords corresponding to u and v .

Proof. Theorem 22 is an immediate consequence of Theorem 21, except for the property that no diagonal of the planar triangulation T of $\mathcal{P}(Z)$ has both its endpoints on one side of \mathcal{P}_G . We sketch only how to construct such a planar triangulation T following the lines of the proof of Theorem 21.

First, for each minimal separator S of H we choose a scanline s such that $S = S(s)$, and this can be done such that no two scanlines cross each other. As mentioned below Theorem 20, none of these scanlines has both endpoints on one side of \mathcal{P}_G . Now we choose all these scanlines as diagonals of a triangulation of $\mathcal{P}(Z)$. If this is not yet a triangulation T of $\mathcal{P}(Z)$ we add more diagonals to obtain a planar triangulation such that we never add a diagonal with both endpoints on one side of \mathcal{P}_G . \square

Consequently, a minimum triangulation H of a k -polygon graph G can be computed by finding a minimum weight triangulation of $\mathcal{P}(Z)$, in which we consider only chords with endpoints on different sides of the polygon. The $O(n^3)$ dynamic programming algorithm for this computation for circle graphs [23] can be adapted to the domain of k -polygon graphs; the adapted algorithm retains its $O(n^3)$ complexity.

It remains to show how to construct a clique tree with at most k leaves for H . This is done by using the planar triangulation T of $\mathcal{P}(Z)$ with $H = H(T)$ for the minimum triangulation H , which is also provided by the algorithm computing H . Now a clique tree of H is constructed as follows. Take the dual graph of the planar triangulation T (without taking a vertex for the exterior face); i.e., each vertex of the dual graph corresponds to a triangle of T . It is well known that this dual graph of a planar triangulation is a tree. Two vertices of the tree are adjacent if and only if the corresponding triangles of T share a diagonal, and we assign to each vertex of the tree the set of all chords intersecting the corresponding triangle of T . This tree has at most k leaves, since any leaf corresponds to a triangle containing a corner of \mathcal{P}_G .

Finally, we remove all nonmaximal cliques by contracting suitable edges of the tree and obtain a clique tree of H with at most k leaves.

THEOREM 23. *There is an $O(n^3)$ algorithm to compute for a given k -polygon graph G a clique tree of a minimum triangulation such that this clique tree has at most k leaves.*

Thus, our approximation algorithm from the previous section applies to this triangulation H of a k -polygon graph G .

Remark. One can show analogously that there is an $O(n^3)$ algorithm that computes for a given minimal triangulation H of a k -polygon graph a clique tree with at most k leaves.

Finally, we combine the main results of this section to obtain an algorithm to approximate the bandwidth of k -polygon graphs.

THEOREM 24. *There is an $O(n^3)$ algorithm to compute for a k -polygon graph G given with a k -polygon representation a layout L satisfying $b(G, L) \leq 2k^2 \cdot bw(G)$.*

Proof. By Theorem 23, there is an $O(n^3)$ algorithm to compute for a k -polygon graph G given with a k -polygon representation, a minimum triangulation H of G , and a clique tree of H such that this clique tree has at most k leaves. By Theorem 13, a layout L satisfying $b(H, L) \leq 2k \cdot bw(H)$ can be computed by a $O(k(n+m))$ algorithm. By Lemma 2, any layout L of a d -good triangulation H of G with $b(H, L) \leq c \cdot bw(H)$ fulfills $b(G, L) \leq d \cdot c \cdot bw(G)$, where $c, d \geq 1$ are constants. Consequently, $b(G, L) \leq 2k^2 \cdot bw(G)$. \square

Remark. By [13], the performance ratio in the previous theorem can be improved to $b(G, L) \leq \frac{4}{3}k^2 \cdot bw(G)$.

REFERENCES

- [1] K. ANDO, A. KANEKO, AND S. GERVAO, *The bandwidth of a tree with k leaves is at most $\lceil \frac{k}{2} \rceil$* , Discrete Math., 150 (1996), pp. 403–406.
- [2] S. F. ASSMANN, G. W. PECK, M. M. SYSŁO, AND J. ZAK, *The bandwidth of caterpillars with hairs of length 1 and 2*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 387–393.
- [3] J. R. S. BLAIR AND B. PEYTON, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, IMA Vol. Math. Appl. 56, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer, New York, 1993, pp. 1–29.
- [4] K. S. BOOTH AND G. S. LUEKER, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.
- [5] P. Z. CHINN, J. CHVÁTALOVÁ, A. K. DEWDNEY, AND N. E. GIBBS, *The bandwidth problem for graphs and matrices—a survey*, J. Graph Theory, 6 (1982), pp. 223–254.
- [6] V. CHVÁTAL, *A remark on a problem of Harary*, Czech Math. J., 20 (1970), pp. 109–111.
- [7] J. CHVÁTALOVÁ, A. K. DEWDNEY, N. E. GIBBS, AND R. R. KORFHAGE, *The Bandwidth Problem for Graphs: A Collection of Recent Results*, Research report #24, Department of Computer Science, University of Western Ontario, London, ON, Canada, 1975.
- [8] D. G. CORNEIL, S. OLARIU, AND L. STEWART, *The ultimate interval graph algorithm?*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, ACM, New York, 1998, pp. 175–180.
- [9] G. A. DIRAC, *On rigid circuit graphs*, Abh. Math. Sem. Univ. Hamburg, 25 (1961), pp. 71–76.
- [10] E. S. ELMALLAH AND L. K. STEWART, *Polygon graph recognition*, J. Algorithms, 26 (1998), pp. 101–140.
- [11] E. M. ESCHEN AND J. P. SPINRAD, *An $O(n^2)$ algorithm for circular-arc graph recognition*, in Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX, 1993, ACM, New York, 1993, pp. 128–137.
- [12] U. FEIGE, *Approximating the bandwidth via volume respecting embeddings*, J. Comput. System Sci., 60 (2000), pp. 510–539.
- [13] F. FOMIN, *private communication*, St. Petersburg, Russia, 1999.

- [14] C. P. GABOR, K. J. SUPOWIT, AND W.-L. HSU, *Recognizing circle graphs in polynomial time*, J. Assoc. Comput. Mach., 36 (1989), pp. 435–473.
- [15] M. R. GAREY, R. L. GRAHAM, D. S. JOHNSON, AND D. E. KNUTH, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math., 34 (1978), pp. 477–495.
- [16] F. GAVRIL, *The intersection graphs of subtrees in a tree are exactly the chordal graphs*, J. Combinatorial Theory Ser. B, 16 (1974), pp. 47–56.
- [17] M. C. GOLUBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [18] S. JIANG, *The Bandwidth Problem and Bandwidth of Cographs*, manuscript, 1992.
- [19] D. J. KLEITMAN AND R. V. VOHRA, *Computing the bandwidth of interval graphs*, SIAM J. Discrete Math., 3 (1990), pp. 373–375.
- [20] T. KLOKS, *Treewidth—Computations and Approximations*, Lecture Notes in Comput. Sci. 842, Springer, Berlin, 1994.
- [21] T. KLOKS, *Treewidth of circle graphs*, Internat. J. Found. Comput. Sci., 7 (1996), pp. 111–120.
- [22] T. KLOKS, D. KRATSCH, AND H. MÜLLER, *Approximating the bandwidth for asteroidal triple-free graphs*, J. Algorithms, 32 (1999), pp. 41–57.
- [23] T. KLOKS, D. KRATSCH, AND C. K. WONG, *Minimum fill-in on circle and circular-arc graphs*, J. Algorithms, 28 (1998), pp. 272–289.
- [24] T. KLOKS AND R.B. TAN, *Bandwidth and topological bandwidth of graphs with few P_4 's*, Discrete Appl. Math., 115 (2001), pp. 117–133.
- [25] R. MAHESH, C. P. RANGAN, AND A. SRINIVASAN, *On finding the minimum bandwidth of interval graphs*, Inform. and Comput., 95 (1991), pp. 218–224.
- [26] B. MONIEN, *The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 505–512.
- [27] C. H. PAPADIMITRIOU, *The NP-completeness of the bandwidth minimization problem*, Computing, 16 (1976), pp. 263–270.
- [28] J. SPINRAD, *Recognition of circle graphs*, J. Algorithms, 16 (1994), pp. 264–282.
- [29] A. P. SPRAGUE, *An $O(n \log n)$ algorithm for bandwidth of interval graphs*, SIAM J. Discrete Math., 7 (1994), pp. 213–220.
- [30] W. UNGER, *The complexity of the approximation of the bandwidth problem*, in Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1998, pp. 82–91.

SCHEDULING UNRELATED MACHINES BY RANDOMIZED ROUNDING*

ANDREAS S. SCHULZ[†] AND MARTIN SKUTELLA[‡]

Abstract. We present a new class of randomized approximation algorithms for unrelated parallel machine scheduling problems with the average weighted completion time objective. The key idea is to assign jobs randomly to machines with probabilities derived from an optimal solution to a linear programming (LP) relaxation in time-indexed variables. Our main results are a $(2+\varepsilon)$ -approximation algorithm for the model with individual job release dates and a $(3/2+\varepsilon)$ -approximation algorithm if all jobs are released simultaneously. We obtain corresponding bounds on the quality of the LP relaxation.

It is an interesting implication for identical parallel machine scheduling that jobs are randomly assigned to machines, in which each machine is equally likely. In addition, in this case the algorithm has running time $O(n \log n)$ and performance guarantee 2. Moreover, the approximation result for identical parallel machine scheduling applies to the on-line setting in which jobs arrive over time as well, with no difference in performance guarantee.

Key words. approximation algorithm, randomized rounding, linear programming relaxation, scheduling, on-line algorithm

AMS subject classifications. 90C27, 68Q25, 90B35, 68M20

PII. S0895480199357078

1. Introduction. It is well known that randomization can help in the design of (approximation) algorithms; cf., e. g., [28, 29]. The use of linear programs (LPs) is one way of guiding randomness. In this paper, we give LP-based, randomized approximation algorithms for parallel machine scheduling problems with the average weighted completion time objective. A randomized ρ -approximation algorithm for a minimization problem is a polynomial-time algorithm that produces for every instance a feasible solution whose expected objective function value is within a factor of ρ of the optimum; ρ is also called the (expected) performance guarantee of the algorithm. Most often, we actually compare the output of an algorithm to a lower bound given by an optimal solution to a certain LP relaxation. Hence, at the same time we obtain a result on the quality of the respective LP. All of our off-line algorithms can be derandomized with no difference in performance guarantees, but at the expense of increased yet still polynomial running times.

We consider the following scheduling model. We are given a set J of n jobs and m unrelated parallel machines. The processing time of job j depends on the machine on which j will be processed; it is a positive integer p_{ij} on machine i . Each job j must be processed for the respective amount of time on one of the m machines and may be assigned to any of them. Every machine can process at most one job at a time. Each job j also has an integral release date $r_j \geq 0$ before which it cannot be started. We denote the completion time of job j in a schedule S by C_j^S ; we also use C_j if no confusion is possible as to which schedule we refer. The objective is to

*Received by the editors May 26, 1999; accepted for publication (in revised form) April 24, 2002; published electronically July 16, 2002. Different parts of this paper appeared in preliminary form in [35, 36].

<http://www.siam.org/journals/sidma/15-4/35707.html>

[†]Massachusetts Institute of Technology, Sloan School of Management, E53-361, 77 Massachusetts Avenue, Cambridge, MA 02139-4307 (schulz@mit.edu).

[‡]Technische Universität Berlin, Fakultät II, Institut für Mathematik, MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany (skutella@math.tu-berlin.de).

minimize the total weighted completion time: a weight $w_j \geq 0$ is associated with each job j , and the goal is to find a schedule S that minimizes $\sum_{j \in J} w_j C_j^S$. The classification scheme introduced by Graham et al. [19] offers a convenient way to refer to individual scheduling problems. The problem that we just described is denoted by $R|r_j|\sum w_j C_j$, and, if all jobs share the same release date, by $R||\sum w_j C_j$. We will also consider the special case of identical parallel machines, which arises by assuming that, for each job j , $p_{ij} = p_j$ for all machines i . In general, we are interested only in nonpreemptive schedules, in which each job must be processed without interruption. Yet, for the identical parallel machine case we also discuss preemptive schedules in which jobs may repeatedly be interrupted and continued later, possibly on a different machine. Hence, the class of problems for which we will present approximation algorithms includes $P||\sum w_j C_j$, $P|r_j, \text{pmtn}|\sum w_j C_j$, and $P|r_j|\sum w_j C_j$. These problems are strongly NP-hard [24, 25].

Scheduling to minimize the total weighted completion time (or, equivalently, the average weighted completion time) has recently received a great deal of attention, partly because of its importance as a classic objective function in scheduling but also because of new applications, for instance, in compiler optimization [6] or in parallel computing [4]. There has been significant progress in the design of approximation algorithms for this class of problems. This progress essentially results from the use of preemptive schedules to construct nonpreemptive ones and from solving an LP relaxation and then constructing a schedule by list scheduling in an order dictated by the LP solution [5, 7, 9, 16, 17, 18, 20, 26, 27, 30, 33, 37, 42].

In this paper, we propose a new technique: random assignments of jobs to machines. In fact, we first introduce an LP relaxation in time-indexed variables for the problem $R|r_j|\sum w_j C_j$, and we then show that a certain variant of randomized rounding leads to an algorithm with performance guarantee 2. If all jobs are released at the same time, $R||\sum w_j C_j$, the performance guarantee of this algorithm is $3/2$. The latter observation was independently made by Chudak [8]. The corresponding LP is a 2-relaxation and a $3/2$ -relaxation, respectively. That is, the true optimum is always within this factor of the optimal value of the LP relaxation. Our algorithm improves upon a $16/3$ -approximation algorithm of Hall et al. [20], which is based on a related interval-indexed LP relaxation. In contrast to their approach, our algorithm does not rely on Shmoys and Tardos's rounding technique for the generalized assignment problem [39]. Rather, we exploit the LP by interpreting the values of the LP variables in an optimal solution as probabilities with which jobs are assigned to machines. For an introduction to randomized rounding and its application to other combinatorial optimization problems, the reader is referred to [28, 34].

Since the time-indexed LP relaxation is of exponential size, we have to resort to an interval-indexed formulation in order to obtain polynomial running times. The resulting algorithm is a $(2 + \varepsilon)$ -approximation algorithm for $R|r_j|\sum w_j C_j$, and a $(3/2 + \varepsilon)$ -approximation algorithm for $R||\sum w_j C_j$, for any $\varepsilon > 0$. The second author subsequently developed a different approach to overcome this difficulty. Based on compact convex quadratic programming relaxations in assignment variables, the same rounding technique yields a 2-approximation algorithm for $R|r_j|\sum w_j C_j$ and a $3/2$ -approximation algorithm for $R||\sum w_j C_j$ directly; see [41].

Actually, for $P|r_j|\sum w_j C_j$, our algorithm produces in time $O(n \log n)$ a solution that is expected to be within a factor of 2 of the optimum. Since the underlying LP relaxation is also a relaxation of the corresponding preemptive problem, this algorithm is a 2-approximation algorithm for $P|r_j, \text{pmtn}|\sum w_j C_j$ as well. The best

TABLE 1.1

Summary of performance guarantees for the minimization of the total weighted completion time. The “Known” columns list the best previously known performance guarantees, whereas the “New” columns list new results from this paper; “—” indicates the absence of a relevant result; and ε is an arbitrarily small, positive constant. While the off-line results are achieved by deterministic algorithms, all on-line results refer to randomized algorithms.

Model	Off-line		On-line	
	Known	New	Known	New
$P \mid r_j \mid \sum C_j$	2.85 [7]	2	$2.89 + \varepsilon$ [5]	2
$P \mid r_j \mid \sum w_j C_j$	$2.89 + \varepsilon$ [5]	2	$2.89 + \varepsilon$ [5]	2
$P \mid r_j, \text{pmtn} \mid \sum C_j$	2 [33]	2	2 [33]	2
$P \mid r_j, \text{pmtn} \mid \sum w_j C_j$	3 [20]	2	—	2
$R \mid \mid \sum w_j C_j$	16/3 [20]	$3/2 + \varepsilon$	—	—
$R \mid r_j \mid \sum w_j C_j$	16/3 [20]	$2 + \varepsilon$	—	—

previously known approximation algorithms had performance guarantees of $(2.89 + \varepsilon)$ and 3, respectively [5, 20]. In addition, our result implies that the value of an optimal nonpreemptive schedule is at most twice the value of an optimal preemptive schedule. Moreover, an optimal solution to the LP used in the case of identical parallel machines is attained by the following preemptive schedule, which can be obtained in a greedy manner. Consider a virtual single machine, which is m times as fast as any of the original machines. At any point in time, schedule from the jobs that are already released, but not yet completed, one with the largest ratio of weight to processing time. We call this schedule the LP schedule. The idea of using a preemptive relaxation on a virtual single machine was employed before by Chekuri et al. [7], among others. They showed that any preemptive schedule on such a machine can be converted into a nonpreemptive schedule on m identical parallel machines such that the completion time of each job j in the nonpreemptive schedule is at most $(3 - 1/m)$ times its preemptive completion time. For the problem of minimizing the average completion time, $P \mid r_j \mid \sum C_j$, they refined this to a 2.85-approximation algorithm. In the single-machine context, the LP schedule is the key ingredient of the 1.6853-approximation algorithm for $1 \mid r_j \mid \sum w_j C_j$ [17] and the 4/3-approximation algorithm for $1 \mid r_j, \text{pmtn} \mid \sum w_j C_j$ [37].

Since an optimal solution to the LP relaxation can be obtained greedily, these single-machine algorithms as well as our algorithm for identical parallel machine scheduling also work in the corresponding on-line setting where jobs continually arrive to be processed, and, for each time t , one must construct the schedule until time t without any knowledge of the jobs that will arrive afterwards; our algorithm maintains an (expected) competitive ratio of 2 for both the nonpreemptive and the preemptive variant of this problem. A randomized on-line algorithm for a minimization problem is ρ -competitive if it outputs for any instance a solution of expected value within a factor of ρ of the value of an optimal off-line solution. A summary of our results, along with a comparison to previously known performance guarantees, is given in Table 1.1.

Recently, Skutella and Woeginger [43] developed a polynomial-time approximation scheme for the problem $P \mid \mid \sum w_j C_j$ that improves upon the previously best known $(1 + \sqrt{2})/2$ -approximation algorithm due to Kawaguchi and Kyan [23]. Subsequently, Afrati et al. [1] gave polynomial-time approximation schemes for the problem $P \mid r_j \mid \sum w_j C_j$, its preemptive variant $P \mid r_j, \text{pmtn} \mid \sum w_j C_j$, and also for the corresponding problems on a constant number of unrelated machines, $Rm \mid r_j \mid \sum w_j C_j$ and $Rm \mid r_j, \text{pmtn} \mid \sum w_j C_j$. On the other hand, Hoogeveen, Schuurman, and Woeg-

inger [22] showed that the problems $R|r_j|\sum C_j$ and $R||\sum w_j C_j$ are APX-hard; hence, they do not possess a polynomial-time approximation scheme, unless $P = NP$.

The remainder of this paper is organized as follows. In section 2, we present our main result: a pseudopolynomial-time algorithm with performance guarantee 2 in the general context of unrelated parallel machine scheduling. We give a combinatorial 2-approximation algorithm for identical parallel machine scheduling in section 3 and show how to use it in an on-line setting. Then, in section 4, we discuss the derandomization of the previously given randomized algorithms. Finally, in section 5, we elaborate on the details of turning the pseudopolynomial-time algorithm of section 2 into a polynomial-time algorithm with performance guarantee $2 + \epsilon$. We conclude by pointing out some open problems in section 6.

2. Scheduling unrelated parallel machines with release dates. In this section, we consider the problem $R|r_j|\sum w_j C_j$. As in [20, 32, 41], we will actually discuss a slightly more general problem, in which the release date of job j may also depend on the machine i to which j is assigned, and is thus denoted by r_{ij} . Machine-dependent release dates are relevant to model situations in which parallel machines are connected by a network; each job is located at a given machine at date 0, and cannot be started on another machine until sufficient time elapses to allow the job to be transmitted to its new machine. This model, called network scheduling, was introduced in [2, 10].

The problem $R|r_{ij}|\sum w_j C_j$ is strongly NP-hard; in fact, $P2||\sum w_j C_j$ is NP-hard, and $1|r_j|\sum C_j$ as well as $P||\sum w_j C_j$ are strongly NP-hard [3, 25]. Phillips, Stein, and Wein presented the first nontrivial approximation algorithm for this problem [32]. It has performance guarantee $O(\log^2 n)$. Subsequently, Hall et al. [20] gave a $16/3$ -approximation algorithm that relies on an interval-indexed LP relaxation whose optimal value serves as a surrogate for the true optimum in their analysis. We use a related LP relaxation and construct feasible schedules from LP solutions by randomized rounding, whereas Hall et al. invoke the deterministic rounding technique of Shmoys and Tardos [39].

Let $T + 1 = \max_{i,j} r_{ij} + \sum_{j \in J} \max_i p_{ij}$ be the time horizon. We introduce for every job $j \in J$, every machine $i = 1, \dots, m$, and every point $t = r_{ij}, \dots, T$ in time a variable y_{ijt} that represents the amount of time job j is processed on machine i within the time interval $(t, t + 1]$. Equivalently, one can say that a y_{ijt}/p_{ij} -fraction of job j is being processed on machine i within the time interval $(t, t + 1]$. The LP relaxation, which is an extension of a single-machine LP relaxation considered by Dyer and Wolsey [11], is as follows:

$$\begin{aligned}
 (2.1) \quad & \min \sum_{j \in J} w_j C_j^{LP} \\
 & \text{s.t.} \quad \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} = 1 \qquad \text{for all } j, \\
 (2.2) \quad & \sum_{j \in J} y_{ijt} \leq 1 \qquad \text{for all } i \text{ and } t, \\
 (2.3) \quad & C_j^{LP} \geq \sum_{i=1}^m \sum_{t=r_{ij}}^T \left(\frac{y_{ijt}}{p_{ij}} \left(t + \frac{1}{2} \right) + \frac{1}{2} y_{ijt} \right) \qquad \text{for all } j,
 \end{aligned}$$

$$(2.4) \quad \begin{aligned} C_j^{LP} &\geq \sum_{i=1}^m \sum_{t=r_{ij}}^T y_{ijt} && \text{for all } j, \\ y_{ijt} &\geq 0 && \text{for all } i, j, \text{ and } t. \end{aligned}$$

We refer to this LP relaxation as (LP_R). Equation (2.1) ensures that the processing requirement of every job is satisfied. The machine capacity constraints (2.2) express that each machine can process at most one job at a time. For (2.3), consider an arbitrary feasible schedule S in which job j is being continuously processed between date $C_j^S - p_{hj}$ and C_j^S on machine h . Then the right-hand side of (2.3) corresponds to the real completion time C_j^S of j if we assign the values to the LP variables y_{ijt} as defined above; i.e., $y_{ijt} = 1$ if $i = h$ and $t \in \{C_j^S - p_{hj}, \dots, C_j^S - 1\}$, and $y_{ijt} = 0$ otherwise. The right-hand side of (2.4) equals the processing time p_{hj} of job j in the schedule S and is therefore a lower bound on its completion time C_j^S . Hence, (LP_R) is a relaxation of the scheduling problem $R \mid r_{ij} \mid \sum w_j C_j$. In fact, even the corresponding mixed-integer program, where the y -variables are forced to be binary, is only a relaxation—it allows preemptions of jobs, and a job may use the capacity of more than one machine at a time.

Due to the exponentially large number of variables, the linear programming relaxation (LP_R) cannot be solved in time polynomial in the input size of an instance of the problem $R \mid r_{ij} \mid \sum w_j C_j$. Therefore, the running time of the following algorithm, which turns an optimal LP solution into a feasible schedule, is only pseudopolynomial. In particular, the results on the quality of the computed schedule that we prove in the remainder of this section do not directly lead to an approximation algorithm for the considered scheduling problem. However, we can overcome this drawback by introducing new variables that are not associated with exponentially many time intervals of length 1 but rather with a polynomial number of intervals of geometrically increasing size. We discuss the technical details of this remedy in section 5.

The following algorithm takes an optimal LP solution and then constructs a feasible schedule by using a variant of randomized rounding.

ALGORITHM LP ROUNDING.

- (1) Compute an optimal solution y to (LP_R).
- (2) For all $j \in J$, assign job j to a machine-time pair (i, t) , where the machine-time pair is chosen from the probability distribution that assigns job j to (i, t) with probability $\frac{y_{ijt}}{p_{ij}}$; set $t_j := t$.
- (3) Schedule on each machine i the jobs that were assigned to it non-preemptively as early as possible in order of nondecreasing t_j ; ties are broken independently at random.

In the analysis of the algorithm, we assume that the random decisions for different jobs in step (2) are pairwise independent.

LEMMA 2.1. *The expected completion time $E[C_j]$ of job j in the schedule constructed by Algorithm LP ROUNDING can be bounded from above by*

$$E[C_j] \leq 2 \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} (t + \frac{1}{2}) + \sum_{i=1}^m \sum_{t=r_{ij}}^T y_{ijt}.$$

If all jobs are released at date 0 on all machines, the following stronger bound holds:

$$(2.5) \quad E[C_j] \leq \sum_{i=1}^m \sum_{t=0}^T \frac{y_{ijt}}{p_{ij}} (t + \frac{1}{2}) + \sum_{i=1}^m \sum_{t=0}^T y_{ijt}.$$

Proof. We start by analyzing the structure of a schedule produced by Algorithm LP ROUNDING. We consider an arbitrary but fixed job $j \in J$ and denote the machine-time pair to which job j has been assigned by (i, t) . Let $\tau \geq 0$ be the earliest point in time such that there is no idle time in the constructed schedule during $(\tau, C_j]$ on machine i . Let K be the set of jobs processed in this time interval on machine i ; hence

$$(2.6) \quad \sum_{k \in K} p_{ik} = C_j - \tau.$$

Since all jobs $k \in K$ are started no later than j , they must have been assigned to machine-time pairs (i, t_k) with $t_k \leq t$. In particular, $r_{ik} \leq t_k \leq t$ for all $k \in K$, and therefore $\tau \leq t$. Together with (2.6) we obtain

$$C_j \leq t + \sum_{k \in K} p_{ik}.$$

To analyze the expected completion time $E[C_j]$ of job j , we first keep the assignment of j to machine-time pair (i, t) fixed and prove a bound on the conditional expectation $E_{i,t}[C_j]$:

$$\begin{aligned} E_{i,t}[C_j] &\leq t + E_{i,t} \left[\sum_{k \in K} p_{ik} \right] \leq t + p_{ij} + \sum_{k \neq j} p_{ik} \cdot \Pr_{i,t}[k \text{ on } i \text{ before } j] \\ &= t + p_{ij} + \sum_{k \neq j} p_{ik} \left(\sum_{\ell=r_{ik}}^{t-1} \frac{y_{ik\ell}}{p_{ik}} + \frac{1}{2} \frac{y_{ikt}}{p_{ik}} \right) \end{aligned}$$

(Note that the factor $\frac{1}{2}$ before the term $\frac{y_{ikt}}{p_{ik}}$ results from breaking ties randomly.)

$$\leq t + p_{ij} + (t + \frac{1}{2}) \leq 2(t + \frac{1}{2}) + p_{ij}.$$

The second but last inequality follows from the machine capacity constraints (2.2). Finally, unconditioning the expectation by the formula of total expectation leads to

$$E[C_j] = \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} E_{i,t}[C_j] \leq 2 \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} (t + \frac{1}{2}) + \sum_{i=1}^m \sum_{t=r_{ij}}^T y_{ijt}.$$

In the absence of nontrivial release dates, we can prove a stronger bound. Observe that $\tau = 0$ and $C_j = \sum_{k \in K} p_{ik}$ in this case. This yields $E_{i,t}[C_j] \leq p_{ij} + (t + \frac{1}{2})$ and eventually the claimed result. \square

THEOREM 2.2. *For instances of $R|r_{ij}|\sum w_j C_j$, the expected objective function value of the schedule constructed by Algorithm LP ROUNDING is at most twice the value of an optimal solution.*

Proof. Lemma 2.1 together with constraints (2.3) implies that the expected completion time of every job j is bounded from above by twice its LP completion time C_j^{LP} . Since the optimal LP value is a lower bound on the value of an optimal schedule and the weights are nonnegative, the result follows from linearity of expectations. \square

Note that Theorem 2.2 still holds if we use the weaker LP relaxation where constraints (2.4) are missing. However, this is not true for the following result.

THEOREM 2.3. *For instances of $R \mid \mid \sum w_j C_j$, Algorithm LP ROUNDING constructs a schedule of expected objective function value at most $3/2$ times the value of an optimal schedule.*

Proof. The claimed result follows from Lemma 2.1 and LP constraints (2.3) and (2.4). \square

In the absence of nontrivial release dates, Algorithm LP ROUNDING can be improved and simplified.

ALGORITHM LP SIMPLE ROUNDING.

- (1) Compute an optimal solution y to (LP_R) .
- (2) For all $j \in J$, assign job j to a machine i , where machine i is chosen from the probability distribution that assigns job j to i with probability $\sum_{t=0}^T \frac{y_{ijt}}{p_{ij}}$.
- (3) Sequence on each machine i the assigned jobs in order of non-increasing ratios w_j/p_{ij} .

Again, the random decisions in step (2) are performed pairwise independently.

COROLLARY 2.4. *For instances of $R \mid \mid \sum w_j C_j$, the approximation result of Theorem 2.3 also holds for Algorithm LP SIMPLE ROUNDING.*

Proof. Notice that the random assignment of jobs to machines is identical in Algorithms LP ROUNDING and LP SIMPLE ROUNDING. Moreover, for a fixed assignment of jobs to machines, sequencing the jobs according to Smith's ratio rule [44] on each machine is optimal. In particular, it improves upon the random sequence used in the final step of Algorithm LP ROUNDING. \square

In the analysis, we have compared the value of the solution computed by Algorithm LP ROUNDING to the optimal LP value, which is a lower bound on the value of an optimal solution. Hence, we obtain the following result on the quality of the LP relaxation.

COROLLARY 2.5. *The linear program (LP_R) is a 2-relaxation for $R \mid r_{ij} \mid \sum w_j C_j$ (even without constraints (2.4)) and a $\frac{3}{2}$ -relaxation for $R \mid \mid \sum w_j C_j$.*

We show in the following section that (LP_R) without constraints (2.4) is not better than a 2-relaxation, even for instances of $P \mid \mid \sum w_j C_j$. On the other hand, the relaxation can be strengthened by adding the constraints

$$(2.7) \quad \sum_{i=1}^m y_{ijt} \leq 1 \quad \text{for } j \in J, t = 0, \dots, T.$$

These constraints ensure that no job can use the capacity of more than one machine in each time period. We do not know whether these constraints can be used to get provably stronger results on the quality of the LP relaxation and better performance guarantees for Algorithm LP ROUNDING.

In section 5, we eventually derive from Theorems 2.2 and 2.3 a $(2 + \varepsilon)$ -approximation algorithm for the problem $R \mid r_{ij} \mid \sum w_j C_j$ and a $(3/2 + \varepsilon)$ -approximation algorithm for $R \mid \mid \sum w_j C_j$.

The techniques presented in this section (and in section 5) can be modified to design approximation algorithms for the corresponding preemptive scheduling problems as well. Notice that, although the LP relaxation (LP_R) allows preemptions of jobs, it is not a relaxation of $R \mid r_{ij}, \text{pmtn} \mid \sum w_j C_j$; one can easily show (see, e.g., [40, Example 2.10.8.]) that the right-hand side of (2.3) can in fact overestimate the actual completion time of a job in the preemptive schedule corresponding to a solution of (LP_R) . However, replacing (2.3) with the following slightly weaker constraint

yields an LP relaxation for the preemptive scheduling problem:

$$C_j^{LP} \geq \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} (t + \frac{1}{2}) \quad \text{for all } j \in J.$$

This leads to a $(3 + \varepsilon)$ -approximation algorithm for $R|r_{ij}, \text{pmtn}|\sum w_j C_j$ and a $(2 + \varepsilon)$ -approximation algorithm for $R|\text{pmtn}|\sum w_j C_j$. These results can again be slightly improved by using convex quadratic programming relaxations; see [41].

In the next section, we consider the special case of identical parallel machines and give a different interpretation of Algorithm LP ROUNDING in terms of so-called α -points. The following variant of Algorithm LP ROUNDING will be useful in this context.

Remark 2.6. The following is an equivalent way of breaking ties randomly in the last step of Algorithm LP ROUNDING: At the end of the second step, draw t_j from the interval $(t, t + 1]$ independently at random with uniform distribution; then, in the last step, ties occur with probability zero and can therefore be neglected.

3. Identical parallel machine scheduling with release dates. We now consider the special case of m *identical* parallel machines. The processing time and the release date of job j no longer depend on the machine job j is assigned to; consequently, they are denoted by p_j and r_j , respectively. As mentioned before, even $P2||\sum w_j C_j$ is NP-hard. We consider $P|r_j|\sum w_j C_j$.

In this context, we can turn Algorithm LP ROUNDING into a purely combinatorial algorithm. Following earlier work (see, e.g., Eastman, Even, and Isaacs [12]), we first reduce an identical parallel machine instance to a single-machine instance. Here, the single machine is assumed to be m times as fast as each of the original m machines; i.e., the processing time of job j on this virtual single machine is $p'_j := p_j/m$. (We assume without loss of generality that p_j is a multiple of m .) Its weight and its release date remain the same. The crucial part of our algorithm is to assign jobs to machines uniformly at random. Then, on each machine, we schedule the assigned jobs in order of random α -points with respect to the LP schedule on the fast single machine.

For $0 < \alpha \leq 1$, the α -point $C_j^S(\alpha)$ of job j with respect to a given preemptive schedule S on the fast single machine is the first point in time when an α -fraction of job j has been completed, i.e., when j has been processed for $\alpha \cdot p'_j$ time units. In particular, $C_j^S(1) = C_j^S$, and for $\alpha = 0$ we define $C_j^S(0)$ to be the starting time of job j . Slightly varying notions of α -points were considered in [21, 33], but their full potential was revealed when Chekuri et al. [7] as well as Goemans [16] chose the parameter α at random. The following procedure may be seen as an extension of their single-machine algorithms to identical parallel machines.

ALGORITHM RANDOM ASSIGNMENT.

- (1) Construct the preemptive LP schedule S on the virtual single machine by scheduling, at any point in time, among the already released but not yet completed jobs the one with largest w_j/p'_j ratio.
- (2) Independently, for all $j \in J$, assign job j to a machine $i \in \{1, \dots, m\}$, where machine i is chosen from the probability distribution that assigns job j to i with probability $\frac{1}{m}$.
- (3) For all $j \in J$, let α_j be a realization of an independent, uniformly distributed random variable in $[0, 1]$.

- (4) Schedule on each machine i the assigned jobs nonpreemptively as early as possible in order of nondecreasing $C_j^S(\alpha_j)$.

Notice that in the first step, whenever a job is released, the job being processed (if any) will be preempted if the released job has a larger w_j/p'_j ratio; here, the p'_j values are the original processing times and not the remaining processing times. The appendix provides an illustration of Algorithm RANDOM ASSIGNMENT. Its running time is dominated by the computation of the preemptive LP schedule in the first step, which can be done in $O(n \log n)$ time using a priority queue [16].

We will show in the following that Algorithm RANDOM ASSIGNMENT can be interpreted as the reformulation of Algorithm LP ROUNDING discussed in Remark 2.6. We notice first that the preemptive LP schedule on the virtual single machine corresponds to an optimal solution to an LP relaxation which is equivalent to (LP_R) . We introduce a variable y_{jt} for every job j and every time period $(t, t + 1]$, $t = r_j, \dots, T$; the variable y_{jt} is set to $1/m$ if job j is being processed on one of the m machines in this period and to 0 otherwise. In contrast to the unrelated parallel machine case, we do not need machine-dependent variables, since it is not necessary to distinguish between the identical parallel machines. We can express the new variables y_{jt} with the help of the old variables y_{ijt} by setting $y_{jt} = \frac{1}{m}(y_{1jt} + \dots + y_{mjt})$ for $j \in J$, $t = r_j, \dots, T$. This leads to the following simplified LP (ignoring constraints (2.4) of (LP_R)):

$$\begin{aligned}
 \min \quad & \sum_{j \in J} w_j C_j^{LP} \\
 \text{s.t.} \quad & \sum_{t=r_j}^T y_{jt} = p'_j && \text{for all } j \in J, \\
 (LP_P) \quad & \sum_{j \in J} y_{jt} \leq 1 && \text{for } t = 0, \dots, T, \\
 & C_j^{LP} = \frac{p_j}{2} + \frac{1}{p'_j} \sum_{t=r_j}^T y_{jt} (t + \frac{1}{2}) && \text{for all } j \in J, \\
 & y_{jt} \geq 0 && \text{for all } j \in J \text{ and } t = r_j, \dots, T.
 \end{aligned}$$

Dyer and Wolsey noticed that this linear program can be solved in $O(n \log n)$ time for the special case $m = 1$ [11]. Goemans showed (also for the case $m = 1$) that the preemptive schedule that is constructed in the first step of Algorithm RANDOM ASSIGNMENT defines an optimal solution to (LP_P) [15]. This result as well as its proof easily generalize to an arbitrary number of identical parallel machines.

LEMMA 3.1. *For instances of $P | r_j | \sum w_j C_j$, the preemptive LP schedule on the fast single machine is an optimal solution to relaxation (LP_P) . Moreover, it can be computed in $O(n \log n)$ time.*

THEOREM 3.2. *RANDOM ASSIGNMENT is a randomized 2-approximation algorithm for $P | r_j | \sum w_j C_j$.*

Proof. We show that Algorithm RANDOM ASSIGNMENT can be interpreted as a special case of the variant of Algorithm LP ROUNDING discussed in Remark 2.6. The result then follows from its polynomial running time and Theorem 2.2.

Lemma 3.1 implies that we compute in the first step of Algorithm RANDOM ASSIGNMENT an optimal solution to the LP relaxation (LP_P) , which is equivalent to (LP_R) without constraints (2.4). In particular, the corresponding solution to (LP_R)

is symmetric with regard to the m machines. Therefore, Algorithm LP ROUNDING assigns each job uniformly at random to one of the machines. The symmetry also yields that for each job j the choice of t_j is not correlated with the choice of i in Algorithm LP ROUNDING.

Next we observe that the probability distributions of the random variable t_j in Algorithm LP ROUNDING and of $C_j^S(\alpha_j)$ in Algorithm RANDOM ASSIGNMENT are the same. In fact, the probability that $C_j^S(\alpha_j) \in (t, t + 1]$ for some t equals the fraction y_{jt}/p'_j of job j that is being processed in this time interval. Moreover, each point in $(t, t + 1]$ is equally likely to be obtained by $C_j^S(\alpha_j)$. Therefore, the random choice of $C_j^S(\alpha_j)$ in Algorithm RANDOM ASSIGNMENT is an alternate way of choosing t_j as described in Algorithm LP ROUNDING. Consequently, the two algorithms coincide for the identical parallel machine case. The result eventually follows from Theorem 2.2. \square

At this point, let us briefly compare the approximation results of this section for the single-machine case ($m = 1$) with related results. If we work only with one α for all jobs instead of individual and independent α_j 's, and if we draw α uniformly from $[0, 1]$, then RANDOM ASSIGNMENT coincides with Goemans's randomized 2-approximation algorithm RANDOM_α for $|r_j| \sum w_j C_j$ [16]. Goemans et al. have improved this result to performance guarantee 1.6853 by using job-dependent α_j 's (as in Algorithm RANDOM ASSIGNMENT) together with a nonuniform choice of the α_j 's [17]. The latter idea can also be applied in the parallel machine setting to obtain a performance guarantee better than 2 for Algorithm RANDOM ASSIGNMENT. However, this improvement depends on m . A comprehensive overview of the use of α -points for machine scheduling problems can be found in [40, Chapter 2].

We have already argued in the previous section that (LP_R) , and thus (LP_P) , is a 2-relaxation of the scheduling problem under consideration.

COROLLARY 3.3. *The relaxation (LP_P) is a 2-relaxation of the scheduling problem $P | r_j | \sum w_j C_j$. This bound is tight, even for $P | | \sum w_j C_j$.*

Proof. The positive result follows from Corollary 2.5. For the tightness of this bound, consider an instance with m machines and one job of length m and unit weight. The optimal LP completion time is $(m + 1)/2$, whereas the optimal completion time is m . When m goes to infinity, the ratio of the two values converges to 2. \square

The following lemma helps to extend Theorem 3.2 and Corollary 3.3 to the corresponding preemptive scheduling problem.

LEMMA 3.4. *Linear program (LP_P) is a relaxation of the preemptive scheduling problem $P | r_j, pmtn | \sum w_j C_j$.*

Proof. Since all release dates and processing times are integral, there exists an optimal preemptive schedule where preemptions only occur at integral points in time. Take such an optimal schedule S and construct the corresponding feasible solution to (LP_P) by setting $y_{jt} = 1/m$ if job j is being processed on one of the m machines within the interval $(t, t + 1]$, and $y_{jt} = 0$ otherwise. We observe that $C_j^{LP} \leq C_j^S$, and equality holds if and only if job j is continuously processed in the interval $(C_j^S - p_j, C_j^S]$. Thus, the value of the constructed solution to (LP_P) is a lower bound on the value of an optimal schedule. \square

COROLLARY 3.5. *For instances of $P | r_j, pmtn | \sum w_j C_j$, the value of the (non-preemptive) schedule produced by Algorithm RANDOM ASSIGNMENT is at most twice the value of an optimal preemptive schedule. Moreover, (LP_P) is a 2-relaxation of this scheduling problem. This bound is tight.*

Another consequence is the following result on the power of preemption.

COROLLARY 3.6. *For identical parallel machine scheduling with release dates so as to minimize the weighted sum of completion times, the value of an optimal nonpreemptive schedule is at most twice the value of an optimal preemptive schedule.*

Moreover, for identical parallel machines, steps two and three of Algorithm LP ROUNDING can be used to convert an arbitrary preemptive schedule into a nonpreemptive one such that the objective function value increases at most by a factor of 2: for a given preemptive schedule, construct the corresponding solution to (LP_R) . The value of this feasible solution to the LP relaxation is a lower bound on the value of the given preemptive schedule. (As discussed at the end of section 2, this is in general not true for unrelated parallel machines.) Using Algorithm LP ROUNDING, the solution to (LP_R) can be turned into a nonpreemptive schedule whose expected value is bounded by twice the value of the LP solution and thus by twice the value of the preemptive schedule we started with. This improves upon a bound of $7/3$ due to Phillips et al. [31].

Several different on-line paradigms have been studied in the area of scheduling; see [38] for a survey. We consider the setting where the only on-line feature is the lack of knowledge of jobs arriving in the future. In particular, the processing time and the weight of a job become known at its arrival. Let us show that Algorithm RANDOM ASSIGNMENT can easily be turned into an on-line algorithm. First, note that we can immediately determine α_j when job j is released; this drawing does not depend on any other decision of the algorithm. The same argument holds for the random machine assignment. Moreover, we can construct the LP schedule until time t without any knowledge of jobs that are released afterwards. Finally, it follows from the analysis in the proof of Lemma 2.1 that we obtain the same performance guarantee if job j is not started before time $t_j = C_j^S(\alpha_j)$. Thus, we modify step four in the on-line variant of Algorithm RANDOM ASSIGNMENT: on each machine we schedule the assigned jobs as early as possible in order of nondecreasing $C_j^S(\alpha_j)$, with the additional constraint that no job j may start before time $C_j^S(\alpha_j)$. This technique was introduced in a similar context in [33].

COROLLARY 3.7. *The on-line variant of Algorithm RANDOM ASSIGNMENT has competitive ratio 2.*

One appealing aspect of Algorithm RANDOM ASSIGNMENT is that the assignment of jobs to machines does not depend on job characteristics; a job is put with probability $1/m$ to any of the machines. This technique also proves useful for the problem without release dates.

THEOREM 3.8. *Assigning jobs independently and uniformly at random to the machines and then applying Smith's ratio rule on each machine is a $3/2$ -approximation algorithm for $P \mid \mid \sum w_j C_j$. There exist instances for which this bound is asymptotically tight.*

Proof. First, notice that the described algorithm is identical to Algorithm RANDOM ASSIGNMENT and therefore to the variant of LP ROUNDING discussed in Remark 2.6. Because of the negative result in Corollary 3.3, we cannot derive the bound of $3/2$ by comparing the expected value of the computed solution to the optimal value of (LP_P) . Remember that we used a stronger relaxation including constraints (2.4) in order to derive this bound in the unrelated parallel machine setting. However, Lemma 2.1 implies that

$$E[C_j] \leq C_j^{LP} + \frac{1}{2}p_j$$

because the second term on the right-hand side of (2.5) is equal to p_j for the case of

identical parallel machines. Since both $\sum_j w_j C_j^{LP}$ and $\sum_j w_j p_j$ are lower bounds on the value of an optimal solution, the result follows.

In order to show that this performance guarantee is tight, we consider instances with m identical parallel machines and m jobs of unit length and weight. We obtain an optimal schedule with value m by assigning one job to each machine. On the other hand, we can show that the expected completion time of a job in the schedule constructed by Algorithm RANDOM ASSIGNMENT is $\frac{3}{2} - \frac{1}{2m}$, which converges to $\frac{3}{2}$ for increasing m . Since the ratio w_j/p_j equals 1 for all jobs, we can without loss of generality schedule on each machine the jobs that were assigned to it in random order. Consider a fixed job j and the machine i it has been assigned to. The probability that a job $k \neq j$ was assigned to the same machine is $1/m$. In this case, job k is processed before j with probability $1/2$. We therefore get $E[C_j] = 1 + \sum_{k \neq j} \frac{1}{2m} = \frac{3}{2} - \frac{1}{2m}$. \square

Interestingly, the derandomized variant of the algorithm considered in Theorem 3.8 coincides with the WSPT-rule: sort the jobs according to nonincreasing ratios w_j/p_j and schedule the next job from this list whenever a machine becomes available. Kawaguchi and Kyan proved that this algorithm has performance guarantee $(1 + \sqrt{2})/2 \approx 1.21$ [23]. While their proof is somewhat intricate, our simpler, probabilistic analysis yields a performance guarantee of $3/2$. However, this weaker result also follows from the work of Eastman, Even, and Isaacs [12]. They gave a combinatorial lower bound for $P \parallel \sum w_j C_j$ that coincides with the lower bound obtained from (LP_P) . The latter observation is due to Uma and Wein [46] and Williamson [49]. Details on the derandomization are given in the next section.

4. Derandomization. The hitherto presented algorithms are randomized and compute feasible schedules whose expected value can be bounded from above. While this shows that our algorithms perform well on average, we cannot give firm guarantees for the performance of a single execution. In certain situations, it may be more desirable to have deterministic algorithms with bounded worst-case ratio. Fortunately, there exists a deterministic version of every algorithm proposed in this paper that maintains the performance guarantee of its randomized companion, as we are about to illustrate now. We can derandomize the randomized algorithms in this paper by using the method of conditional probabilities. The method of conditional probabilities is one of the most important techniques for derandomization. This method is implicitly contained in a paper of Erdős and Selfridge [14] and was extended to a more general context by Spencer [45]. It considers the random decisions one after the other and chooses the most promising alternative at every decision point. Here, it is assumed that all remaining decisions are random. Thus, an alternative is said to be most promising if the corresponding conditional expected objective function value is smallest. We shall demonstrate this technique for the most general problem, $R|r_{ij} \parallel \sum w_j C_j$, and Algorithm LP ROUNDING.

Our analysis of Algorithm LP ROUNDING in the proof of Lemma 2.1 does not give a precise expression for the expected value of the computed solution but only an upper bound. Hence, we modify Algorithm LP ROUNDING by replacing its last step with the following variant:

- (3') Schedule on each machine i the assigned jobs nonpreemptively in order of nondecreasing t_j ; ties are broken by preferring jobs with smaller indices. At the starting time of job j , the amount of idle time on its machine has to be exactly t_j .

Since $r_{ij} \leq t_j$ for each job j that has been assigned to machine i and $t_j \leq t_k$ if job k is scheduled after job j , step (3') defines a feasible schedule. In the proof of

Lemma 2.1, we have bounded the idle time before the start of job j from above by t_j . Thus, the analysis still works for the modified version of Algorithm LP ROUNDING. The advantage of the modification is that we are now in a position to give precise expressions for the expectations and conditional expectations of completion times.

Let y be an optimal solution to (LP_R) . Using the same arguments as in the proof of Lemma 2.1, we obtain the following expression for the expected completion time of job j in the schedule output by the modified Algorithm LP ROUNDING:

$$E[C_j] = \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} \left(p_{ij} + t + \sum_{k \neq j} \sum_{\ell=r_{ik}}^{t-1} y_{ik\ell} + \sum_{k < j} y_{ikt} \right).$$

Moreover, we are also interested in the conditional expectation of j 's completion time if some of the jobs have already been assigned to a machine-time pair. Let $K \subseteq J$ be such a subset of jobs. For each job $k \in K$, let the 0/1-variable x_{ikt} for $t \geq r_{ik}$ indicate whether k has been assigned to the machine-time pair (i, t) (i.e., $x_{ikt} = 1$) or not ($x_{ikt} = 0$). This enables us to give the following expressions for the conditional expectation of j 's completion time. If $j \notin K$, we have

$$(4.1) \quad E_{K,x}[C_j] = \sum_{i=1}^m \sum_{t=r_{ij}}^T \frac{y_{ijt}}{p_{ij}} \left(p_{ij} + t + \sum_{k \in K} \sum_{\ell=r_{ik}}^{t-1} x_{ik\ell} p_{ik} + \sum_{k \in K, k < j} x_{ikt} p_{ik} + \sum_{k \in J \setminus (K \cup \{j\})} \sum_{\ell=r_{ik}}^{t-1} y_{ik\ell} + \sum_{k \in J \setminus K, k < j} y_{ikt} \right),$$

and, if $j \in K$, we obtain

$$(4.2) \quad E_{K,x}[C_j] = p_{ij} + t + \sum_{k \in K} \sum_{\ell=r_{ik}}^{t-1} x_{ik\ell} p_{ik} + \sum_{k \in K, k < j} x_{ikt} p_{ik} + \sum_{k \in J \setminus K} \sum_{\ell=r_{ik}}^{t-1} y_{ik\ell} + \sum_{k \in J \setminus K, k < j} y_{ikt},$$

where (i, t) is the machine-time pair job j has been assigned to, i.e., $x_{ijt} = 1$. The following lemma is the most important part of derandomizing Algorithm LP ROUNDING.

LEMMA 4.1. *Let y be an optimal solution to (LP_R) , $K \subseteq J$, and let x be a fixed assignment of the jobs in K to machine-time pairs. Furthermore, let $j \in J \setminus K$. Then there exists an assignment of j to a machine-time pair (i, t) (i.e., $x_{ijt} = 1$) with $r_{ij} \leq t$ such that*

$$(4.3) \quad E_{K \cup \{j\}, x} \left[\sum_{\ell} w_{\ell} C_{\ell} \right] \leq E_{K,x} \left[\sum_{\ell} w_{\ell} C_{\ell} \right].$$

Proof. Using the formula of total expectation, the conditional expectation on the right-hand side of (4.3) can be written as a convex combination of conditional expectations $E_{K \cup \{j\}, x} [\sum_{\ell} w_{\ell} C_{\ell}]$ over all possible assignments of job j to machine-time pairs (i, t) with coefficients $\frac{y_{ijt}}{p_{ij}}$. The best one satisfies (4.3). \square

Lemma 4.1 leads to a derandomized version of Algorithm LP ROUNDING if we replace the second step by the following variant:

- (2') Set $K := \emptyset$; $x := 0$; for all $j \in J$ do
 - (i) for all possible assignments of j to machine-time pairs (i, t) (i.e., $x_{ijt} = 1$) compute $E_{K \cup \{j\}, x}[\sum_{\ell} w_{\ell} C_{\ell}]$;
 - (ii) determine the machine-time pair (i, t) that minimizes the conditional expectation in (i);
 - (iii) set $K := K \cup \{j\}$; $x_{ijt} = 1$.

Notice that we have replaced step (3) of Algorithm LP ROUNDING by (3') only to give a more accessible analysis of its derandomization. Since the value of the schedule constructed in step (3) is always at least as good as the one constructed in step (3'), the following theorem can be formulated for Algorithm LP ROUNDING with the original step (3).

THEOREM 4.2. *If we replace step (2) in Algorithm LP ROUNDING with (2'), we obtain a deterministic algorithm with performance guarantee 2 for $R \mid r_{ij} \mid \sum w_j C_j$ and with performance guarantee 3/2 for $R \mid \mid \sum w_j C_j$. Moreover, the running time of this algorithm is polynomial in the number of variables of (LP_R) .*

Proof. The result follows by an inductive use of Lemma 4.1 and from Theorems 2.2 and 2.3. The computation of (4.1) and (4.2) is polynomially bounded by the number of variables. Therefore, the running time of each of the n iterations in step (2') is polynomially bounded by this number as well. \square

The same derandomization process also works for the polynomial-time approximation algorithms in section 5 that are based on interval-indexed LP relaxations. Since these LP relaxations contain only a polynomial number of variables, the running time of the derandomized algorithms is polynomially bounded in the input size of the scheduling problem.

The derandomization of Algorithm RANDOM ASSIGNMENT for $P \mid \mid \sum w_j C_j$ by the method of conditional probabilities leads to an interesting result, as indicated at the end of section 3. It essentially follows from the considerations above that the derandomized version of this algorithm assigns each job to the machine with the smallest load. If we consider the jobs in order of nonincreasing ratios w_j/p_j , the resulting algorithm coincides with the WSPT-rule.

5. Interval-indexed LP relaxations. We pointed out earlier that the LP-based Algorithms LP ROUNDING and LP SIMPLE ROUNDING for unrelated parallel machine scheduling suffer from the exponential number of variables in the corresponding LP relaxation (LP_R) . However, we can overcome this drawback by using new variables that are not associated with exponentially many time intervals of length 1 but with a polynomial number of intervals of geometrically increasing size. This idea was earlier introduced by Hall, Shmoys, and Wein [21]. We show that Algorithm LP ROUNDING can be turned into a polynomial-time algorithm for $R \mid r_{ij} \mid \sum w_j C_j$ at the cost of an increased performance guarantee of $2 + \varepsilon$. The same technique can be used to modify Algorithm LP SIMPLE ROUNDING to a $(3/2 + \varepsilon)$ -approximation algorithm for $R \mid \mid \sum w_j C_j$.

For a given $\eta > 0$, the number L is chosen to be the smallest integer such that $(1 + \eta)^L \geq T + 1$. Consequently, L is polynomially bounded in the input size of the considered scheduling problem. Let $I_0 = [0, 1]$ and for $1 \leq \ell \leq L$ let $I_{\ell} = ((1 + \eta)^{\ell-1}, (1 + \eta)^{\ell}]$. We denote with $|I_{\ell}|$ the length of the ℓ th interval; i.e., $|I_{\ell}| = \eta(1 + \eta)^{\ell-1}$ for $1 \leq \ell \leq L$. To simplify notation we define $(1 + \eta)^{\ell-1}$ to be $\frac{1}{2}$ for $\ell = 0$. We introduce variables $y_{ij\ell}$ for $i = 1, \dots, m, j \in J$, and $(1 + \eta)^{\ell-1} \geq r_{ij}$ with the following interpretation: $y_{ij\ell} \cdot |I_{\ell}|$ is the time job j is processed on machine i within time interval I_{ℓ} , or, equivalently, $(y_{ij\ell} \cdot |I_{\ell}|)/p_j$ is the fraction of job j that is being

processed on machine i within I_ℓ . Consider the following linear program in these interval-indexed variables:

$$\begin{aligned}
 (5.1) \quad & \min \sum_{j \in J} w_j C_j \\
 \text{s. t.} \quad & \sum_{i=1}^m \sum_{\substack{\ell=0 \\ (1+\eta)^{\ell-1} \geq r_{ij}}}^L \frac{y_{ij\ell} \cdot |I_\ell|}{p_{ij}} = 1 && \text{for all } j, \\
 (5.2) \quad & \sum_{j \in J} y_{ij\ell} \leq 1 && \text{for all } i \text{ and } \ell, \\
 (5.3) \quad & C_j = \sum_{i=1}^m \sum_{\substack{\ell=0 \\ (1+\eta)^{\ell-1} \geq r_{ij}}}^L \left(\frac{y_{ij\ell} \cdot |I_\ell|}{p_{ij}} (1+\eta)^{\ell-1} + \frac{1}{2} \cdot y_{ij\ell} \cdot |I_\ell| \right) && \text{for all } j, \\
 & y_{ij\ell} \geq 0 && \text{for all } i, j, \text{ and } \ell.
 \end{aligned}$$

We refer to this LP relaxation as (LP_R^η) .

Consider a feasible schedule and assign the values to the variables $y_{ij\ell}$ as defined above. This solution is clearly feasible: constraints (5.1) are satisfied since a job j consumes p_{ij} time units if it is processed on machine i ; constraints (5.2) are satisfied since the total processing time of jobs that are processed within the interval I_ℓ on machine i cannot exceed its length. Finally, if job j is continuously being processed between $C_j - p_{hj}$ and C_j on machine h , then the right-hand side of (5.3) is a lower bound on the real completion time. Thus, (LP_R^η) is a relaxation of the scheduling problem $R|r_{ij}|\sum w_j C_j$. Since (LP_R^η) is of polynomial size, an optimal solution can be computed in polynomial time. We rewrite Algorithm LP ROUNDING for the new LP:

ALGORITHM LP ROUNDING.

- (1) Compute an optimal solution y to (LP_R^η) .
- (2) Independently, for all $j \in J$, assign job j to a machine-interval pair (i, I_ℓ) , where the machine-interval pair is chosen from the probability distribution that assigns job j to (i, I_ℓ) with probability $\frac{y_{ij\ell} \cdot |I_\ell|}{p_{ij}}$; set t_j to the left endpoint $(1+\eta)^{\ell-1}$ of the time interval I_ℓ .
- (3) Schedule on each machine i the assigned jobs in order of nondecreasing t_j ; ties are randomly broken.

THEOREM 5.1. *The expected completion time of each job j in the schedule constructed by Algorithm LP ROUNDING is at most $2 \cdot (1+\eta) \cdot C_j^{LP}$.*

Proof. We argue almost exactly as in the proof of Lemma 2.1. We consider an arbitrary but fixed job $j \in J$. We also consider a fixed assignment of j to machine i and time interval I_ℓ . Again, the conditional expectation of j 's starting time equals the expected idle time plus the expected processing time on machine i before j is started.

With similar arguments as in the proof of Lemma 2.1, we can bound the sum of the idle time plus the processing time by $2 \cdot (1+\eta) \cdot (1+\eta)^{\ell-1}$. This, together with the expected processing time of job j itself, which is $\sum_{i=1}^m \sum_{\ell=0}^L y_{ij\ell} \cdot |I_\ell|$, and (5.3), yields the theorem. \square

For any given $\varepsilon > 0$, we can choose $\eta = \varepsilon/2$. Then Algorithm LP ROUNDING is a $(2 + \varepsilon)$ -approximation algorithm for the problem $R|r_{ij}|\sum w_j C_j$, and (LP_R^η) is a $(2 + \varepsilon)$ -relaxation.

6. Concluding remarks and open problems. In this paper, we have developed LP-based approximation algorithms for a variety of parallel machine scheduling problems with the average weighted completion time objective. A by-product of our analysis are results on the quality of the underlying LP relaxations.

Our central off-line result is the $(2 + \varepsilon)$ -approximation algorithm for the problem $R|r_{ij}|\sum w_j C_j$, and there exist instances which show that the underlying LP relaxation $((LP_R)$ without inequalities (2.4)) is indeed not better than a 2-relaxation. However, it is open whether the quality of (LP_R) (with (2.4) and/or (2.7)) is better than 2 and also whether it can be used to derive an approximation algorithm with performance guarantee strictly less than 2. On the negative side, $R|r_j|\sum C_j$ is APX-hard [22]. In other words, the best known approximation algorithm for $R|r_{ij}|\sum w_j C_j$ has performance guarantee 2 (we proved $2 + \varepsilon$ here, and [41] gets rid of the ε using a convex quadratic relaxation), but the only known limit to its approximation is the nonexistence of a polynomial-time approximation scheme (PTAS), unless $P = NP$. The situation for $R||\sum w_j C_j$ is similar. (LP_R) is a $3/2$ -relaxation, the quality of (LP_R) together with (2.7) is unknown, the $3/2$ -approximation given in [41] (improving upon the $(3/2 + \varepsilon)$ -approximation in section 2) is best known, and again there cannot be a PTAS, unless $P = NP$ [22]. For identical parallel machines, one important property of our 2-approximation algorithm for $P|r_j|\sum w_j C_j$ is that it runs in time $O(n \log n)$. The running time of the recent PTAS is $O((m+1)^{poly(1/\varepsilon)}n + n \log n)$ [1]. The other important feature of the $O(n \log n)$ algorithm is that it is capable of working in an on-line context as well, which brings us to the second set of open problems.

If jobs arrive over time and if the performance of algorithms is measured in terms of their competitiveness to optimal off-line algorithms, it is important to distinguish between deterministic and randomized algorithms. For identical parallel machine scheduling, there is a significant gap between the best known lower bound and competitive ratio of a deterministic algorithm. A universal lower bound of 1.309 is proved in [47, Chapter 3], while a $(4 + \varepsilon)$ -competitive algorithm emerges from a more general framework given in [20]. For randomized algorithms, the situation is only slightly better. No relevant lower bound on the competitive ratio of any randomized on-line algorithm is known, and the modified version of Algorithm RANDOM ASSIGNMENT discussed in section 3 is a randomized 2-competitive algorithm.

An interesting application of the approximation results for $R|r_{ij}|\sum w_j C_j$ and $R||\sum w_j C_j$ was subsequently proposed in [13]. In a generalization of standard scheduling models, Engels et al. introduce the possibility to outsource a job j at a cost e_j . Since there is an approximation-preserving reduction from an instance of the resulting scheduling with rejection problem $R|(r_j)|\sum_S w_j C_j + \sum_{\bar{S}} e_j$ to an instance of $R|(r_{ij})|\sum w_j C_j$, the approximation results are inherited.

Finally, in a computational study of $R||\sum w_j C_j$, Vredeveld and Hurkens [48] compared the algorithms based on time-indexed and interval-indexed formulations proposed herein to the algorithm based on a convex quadratic programming relaxation in [41] and to a variant based on a time-indexed LP that uses variables x_{ijt} . Here, $x_{ijt} = 1$ if and only if job j is started at time t on machine i . The latter relaxation is tighter than (LP_R) and the convex program in [41] (see, e.g., [48]), and this is confirmed by their studies. In addition, the algorithm based on the relaxation in x -

variables also leads to the best upper bounds on their test instances. It follows from Theorem 2.3 that randomized rounding based on this relaxation has performance guarantee $3/2$ as well.

Appendix. An illustrating example. Consider the following instance of $P2|r_j|\sum w_j C_j$, consisting of the job set $\{1, 2, 3, 4\}$ together with fixed values α_j :





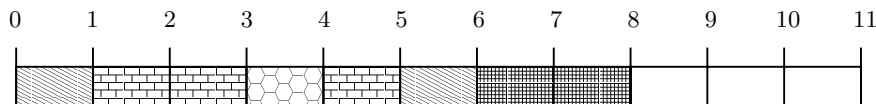
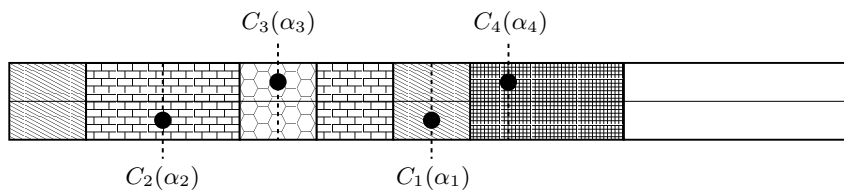
job j	r_j	p_j	w_j/p_j	α_j	
1		0	4	1	$3/4$
2		1	6	2	$1/3$
3		3	2	3	$1/2$
4		6	4	2	$1/4$

Figure A.1 illustrates the action of Algorithm RANDOM ASSIGNMENT for the given instance. In the first step, it computes the preemptive LP schedule S on a virtual single machine, which is twice as fast as each of the original two machines. Then each job is randomly assigned to one of the two machines, and the α -points are chosen. Finally, the jobs are scheduled on their machines nonpreemptively in nondecreasing order of $C_j^S(\alpha_j)$.

(1) preemptive schedule:



(2) & (3) random choices:



(4) feasible schedule:

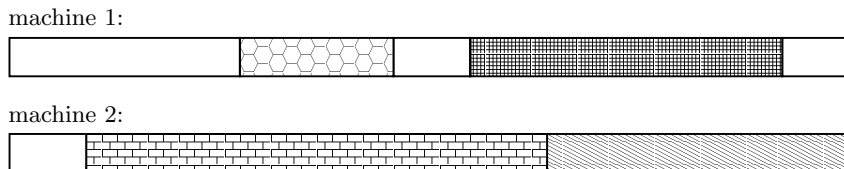


FIG. A.1.

Acknowledgments. The authors are grateful to Chandra S. Chekuri, Michel X. Goemans, and David B. Shmoys for helpful comments on an earlier version of this paper [36]. They would also like to thank two anonymous referees for their careful reading of the manuscript and numerous helpful comments that helped to improve the presentation of this paper.

REFERENCES

- [1] F. AFRATI, E. BAMPIS, C. CHEKURI, D. KARGER, C. KENYON, S. KHANNA, I. MILIS, M. QUEYRANNE, M. SKUTELLA, C. STEIN, AND M. SVIRIDENKO, *Approximation schemes for minimizing average weighted completion time with release dates*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, New York, 1999, pp. 32–43.
- [2] B. AWERBUCH, S. KUTTEN, AND D. PELEG, *Competitive distributed job scheduling*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, Victoria, BC, 1992, pp. 571–581.
- [3] J. L. BRUNO, E. G. COFFMAN, JR., AND R. SETHI, *Scheduling independent tasks to reduce mean finishing time*, Comm. ACM, 17 (1974), pp. 382–387.
- [4] S. CHAKRABARTI AND S. MUTHUKRISHNAN, *Resource scheduling for parallel database and scientific applications*, in Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures, Padua, Italy, 1996, pp. 329–335.
- [5] S. CHAKRABARTI, C. PHILLIPS, A. S. SCHULZ, D. B. SHMOYS, C. STEIN, AND J. WEIN, *Improved scheduling algorithms for minsum criteria*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1099, F. Meyer auf der Heide and B. Monien, eds., Springer, Berlin, 1996, pp. 646–657.
- [6] C. S. CHEKURI, R. JOHNSON, R. MOTWANI, B. NATARAJAN, B. R. RAU, AND M. SCHLANSKER, *Profile-driven instruction level parallel scheduling with applications to super blocks*, in Proceedings of the 29th Annual International Symposium on Microarchitecture, Paris, France, 1996, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 58–67.
- [7] C. CHEKURI, R. MOTWANI, B. NATARAJAN, AND C. STEIN, *Approximation techniques for average completion time scheduling*, SIAM J. Comput., 31 (2001), pp. 146–166.
- [8] F. A. CHUDAK, *A min-sum 3/2-approximation algorithm for scheduling unrelated parallel machines*, J. Sched., 2 (1999), pp. 73–77.
- [9] F. A. CHUDAK AND D. B. SHMOYS, *Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds*, J. Algorithms, 30 (1999), pp. 323–343.
- [10] X. DENG, H. LIU, J. LONG, AND B. XIAO, *Deterministic load balancing in computer networks*, in Proceedings of the 2nd Annual IEEE Symposium on Parallel and Distributed Processing, Dallas, TX, 1990, pp. 50–57.
- [11] M. E. DYER AND L. A. WOLSEY, *Formulating the single machine sequencing problem with release dates as a mixed integer program*, Discrete Appl. Math., 26 (1990), pp. 255–270.
- [12] W. L. EASTMAN, S. EVEN, AND I. M. ISAACS, *Bounds for the optimal scheduling of n jobs on m processors*, Management Sci., 11 (1964), pp. 268–279.
- [13] D. W. ENGELS, D. R. KARGER, S. G. KOLLIPOULOS, S. SENGUPTA, R. N. UMA, AND J. WEIN, *Techniques for scheduling with rejection*, in Algorithms—ESA '98, Lecture Notes in Comput. Sci. 1461, G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, eds., Springer, Berlin, 1998, pp. 490–501.
- [14] P. ERDŐS AND J. L. SELFRIDGE, *On a combinatorial game*, J. Combinatorial Theory Ser. A, 14 (1973), pp. 298–301.
- [15] M. X. GOEMANS, *A supermodular relaxation for scheduling with release dates*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer, Berlin, 1996, pp. 288–300.
- [16] M. X. GOEMANS, *Improved approximation algorithms for scheduling with release dates*, in Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, ACM, New York, 1997, pp. 591–598.
- [17] M. X. GOEMANS, M. QUEYRANNE, A. S. SCHULZ, M. SKUTELLA, AND Y. WANG, *Single machine scheduling with release dates*, SIAM J. Discrete Math., 15 (2002), pp. 165–192.
- [18] M. X. GOEMANS, J. WEIN, AND D. P. WILLIAMSON, *A 1.47-approximation algorithm for a preemptive single-machine scheduling problem*, Oper. Res. Lett., 26 (2000), pp. 149–154.
- [19] R. L. GRAHAM, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Ann. Discrete Math., 5 (1979), pp. 287–326.
- [20] L. A. HALL, A. S. SCHULZ, D. B. SHMOYS, AND J. WEIN, *Scheduling to minimize average completion time: Off-line and on-line approximation algorithms*, Math. Oper. Res., 22 (1997), pp. 513–544.
- [21] L. A. HALL, D. B. SHMOYS, AND J. WEIN, *Scheduling to minimize average completion time: Off-line and on-line algorithms*, in Proceedings of the Seventh Annual ACM-SIAM Sym-

- posium on Discrete Algorithms, Atlanta, GA, 1996, ACM, New York, 1996, pp. 142–151.
- [22] H. HOOGEVEEN, P. SCHURMAN, AND G. J. WOEGINGER, *Non-approximability results for scheduling problems with minsum criteria*, *INFORMS J. Comput.*, 13 (2001), pp. 157–168.
- [23] T. KAWAGUCHI AND S. KYAN, *Worst case bound of an LRF schedule for the mean weighted flow-time problem*, *SIAM J. Comput.*, 15 (1986), pp. 1119–1129.
- [24] J. LABETOULLE, E. L. LAWLER, J. K. LENSTRA, AND A. H. G. RINNOOY KAN, *Preemptive scheduling of uniform machines subject to release dates*, in *Progress in Combinatorial Optimization*, W. R. Pulleyblank, ed., Academic Press, New York, 1984, pp. 245–261.
- [25] J. K. LENSTRA, A. H. G. RINNOOY KAN, AND P. BRUCKER, *Complexity of machine scheduling problems*, *Ann. Discrete Math.*, 1 (1977), pp. 343–362.
- [26] R. H. MÖHRING, M. W. SCHÄFFTER, AND A. S. SCHULZ, *Scheduling jobs with communication delays: Using infeasible solutions for approximation*, in *Algorithms—ESA '96*, Lecture Notes in Comput. Sci. 1136, J. Diaz and M. Serna, eds., Springer, Berlin, 1996, pp. 76–90.
- [27] R. H. MÖHRING, A. S. SCHULZ, AND M. UETZ, *Approximation in stochastic scheduling: The power of LP-based priority policies*, *J. ACM*, 46 (1999), pp. 924–942.
- [28] R. MOTWANI, J. NAOR, AND P. RAGHAVAN, *Randomized approximation algorithms in combinatorial optimization*, in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, ed., Thomson, Boston, MA, 1996, pp. 447–481.
- [29] R. MOTWANI AND P. RAGHAVAN, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.
- [30] A. MUNIER, M. QUEYRANNE, AND A. S. SCHULZ, *Approximation bounds for a general class of precedence constrained parallel machine scheduling problems*, in *Integer Programming and Combinatorial Optimization*, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Springer, Berlin, 1998, pp. 367–382.
- [31] C. PHILLIPS, A. S. SCHULZ, D. B. SHMOYS, C. STEIN, AND J. WEIN, *Improved bounds on relaxations of a parallel machine scheduling problem*, *J. Comb. Optim.*, 1 (1998), pp. 413–426.
- [32] C. PHILLIPS, C. STEIN, AND J. WEIN, *Task scheduling in networks*, *SIAM J. Discrete Math.*, 10 (1997), pp. 573–598.
- [33] C. PHILLIPS, C. STEIN, AND J. WEIN, *Minimizing average completion time in the presence of release dates*, *Math. Programming*, 82 (1998), pp. 199–223.
- [34] P. RAGHAVAN AND C. D. THOMPSON, *Randomized rounding: A technique for provably good algorithms and algorithmic proofs*, *Combinatorica*, 7 (1987), pp. 365–374.
- [35] A. S. SCHULZ AND M. SKUTELLA, *Random-based scheduling: New approximations and LP lower bounds*, in *Randomization and Approximation Techniques in Computer Science*, Lecture Notes in Comput. Sci. 1269, J. Rolim, ed., Springer, Berlin, 1997, pp. 119–133.
- [36] A. S. SCHULZ AND M. SKUTELLA, *Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria*, in *Algorithms—ESA '97*, Lecture Notes in Comput. Sci. 1284, R. Burkard and G. J. Woeginger, eds., Springer, Berlin, 1997, pp. 416–429.
- [37] A. S. SCHULZ AND M. SKUTELLA, *The power of α -points in preemptive single machine scheduling*, *J. Sched.*, 5 (2002), pp. 121–133.
- [38] J. SGALL, *On-line scheduling—a survey*, in *Online Algorithms: The State of the Art*, Lecture Notes in Comput. Sci. 1442, A. Fiat and G. J. Woeginger, eds., Springer, Berlin, 1998, pp. 196–231.
- [39] D. B. SHMOYS AND E. TARDOS, *An approximation algorithm for the generalized assignment problem*, *Math. Programming*, 62 (1993), pp. 461–474.
- [40] M. SKUTELLA, *Approximation and Randomization in Scheduling*, Ph.D. thesis, Technische Universität Berlin, Berlin, Germany, 1998.
- [41] M. SKUTELLA, *Convex quadratic and semidefinite programming relaxations in scheduling*, *J. ACM*, 48 (2001), pp. 206–242.
- [42] M. SKUTELLA AND M. UETZ, *Scheduling precedence-constrained jobs with stochastic processing times on parallel machines*, in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001, ACM, New York, 2001, pp. 589–590.
- [43] M. SKUTELLA AND G. J. WOEGINGER, *A PTAS for minimizing the total weighted completion time on identical parallel machines*, *Math. Oper. Res.*, 25 (2000), pp. 63–75.
- [44] W. E. SMITH, *Various optimizers for single-stage production*, *Naval Res. Logist. Quart.*, 3 (1956), pp. 59–66.
- [45] J. SPENCER, *Ten Lectures on the Probabilistic Method*, CBMS-NSF Reg. Conf. Ser. in Appl. Math. 52, SIAM, Philadelphia, 1987.
- [46] R. N. UMA AND J. M. WEIN, *On the relationship between combinatorial and LP-based approaches to NP-hard scheduling problems*, in *Integer Programming and Combinatorial Op-*

- timization, Lecture Notes in Comput. Sci. 1412, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Springer, Berlin, 1998, pp. 394–408.
- [47] A. P. A. VESTJENS, *On-Line Machine Scheduling*, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, 1997.
- [48] T. VREDEVELD AND C. HURKENS, *Experimental Comparison of Approximation Algorithms for Scheduling Unrelated Parallel Machines*, SPOR report, Eindhoven University of Technology, The Netherlands, 2001.
- [49] D. P. WILLIAMSON, Cited as private communication in [46], 1997.

ON THE FACET-INDUCING ANTIWEB-WHEEL INEQUALITIES FOR STABLE SET POLYTOPES*

EDDIE CHENG[†] AND SVEN DE VRIES[‡]

Abstract. A large class of facets is constructed for the stable set polytope. This class is a common generalization of wheel facets and of antiweb facets. The proof of their validity and facetness exploits graph operations which transform inequalities into more complicated ones. In an accompanying paper polynomial time separation-algorithms are presented for generalizations of these inequalities.

Key words. stable sets, polyhedral combinatorics, cutting planes, facets, wheels, antiwebs

AMS subject classifications. 90C10, 90C27, 90C35, 90C57

PII. S0895480101391053

1. Introduction. Let $G = (V, E)$ be a simple connected graph with $|V| = n \geq 2$ and $|E| = m$. A subset of V is called a *stable set* if it does not contain adjacent vertices of G . Given weights for all vertices, the *stable set problem* is now to find a stable set of maximum weight. Let $N \subseteq V$. The *incidence vector* of N is $\chi^N \in \{0, 1\}^V$ such that $\chi_v^N = 1$ if and only if $v \in N$. The *stable set polytope* of G , denoted by $\text{STAB}(G)$, is the convex hull of incidence vectors of stable sets of G . Some well-known valid inequalities for $\text{STAB}(G)$ (that is, the inequalities are satisfied by all points in $\text{STAB}(G)$) include the *trivial inequalities* ($x_v \geq 0$ for $v \in V$), the *cycle inequalities* ($\sum_{v \in C} x_v \leq k$, where C is the vertex-set of a cycle of length $2k + 1$), and the *clique inequalities* ($\sum_{v \in K} x_v \leq 1$, where K induces a clique). A clique inequality is called an *edge inequality* if the clique has just two vertices. Another important class of inequalities are the *antiweb inequalities* which generalize cycle and clique inequalities.

A valid inequality $a^\top x \leq \alpha$ is *facet-inducing* for $\text{STAB}(G)$ if $\{x : a^\top x = \alpha\} \cap \text{STAB}(G)$ has dimension one less than that of $\text{STAB}(G)$. We start our study of facetness in section 3 by describing three operations—adding an apex, doubly subdividing an edge, and applying star-subdivision—that can transform facets into facets; furthermore, their interaction is studied in section 4. The knowledge about these operations is applied in section 5 to introduce a large new class of valid inequalities—the antiweb- s -wheel inequalities—and to completely characterize the facet-inducing inequalities among all proper antiweb- s -wheel inequalities. Finally, we provide a brief view into questions of facetness for improper antiweb-wheels.

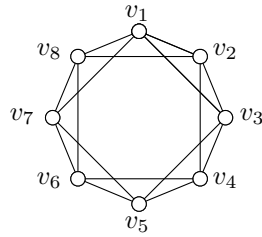
2. Antiwebs. Let n and t be integers such that $t \geq 2$, $n \geq 2t - 1$, and $n \not\equiv 0 \pmod{t}$. An (n, t) -*antiweb* \mathcal{AW} is a graph with vertex-set $\{v_1, v_2, \dots, v_n\}$; two vertices v_i and v_j ($i > j$) are adjacent if $k := \min\{i - j, n + j - i\} \leq t - 1$; we call $\{v_i, v_j\}$ a *cross-edge* of *type* k , or simply *k-edge*. A 1-edge may also be referred to as

*Received by the editors June 19, 2001; accepted for publication (in revised form) June 6, 2002; published electronically September 10, 2002.

<http://www.siam.org/journals/sidma/15-4/39105.html>

[†]Department of Mathematics and Statistics, Oakland University, Rochester, MI 48309 (echeng@oakland.edu). This author's research was partially supported by a faculty research fellowship from the Oakland University Foundation.

[‡]Zentrum Mathematik, TU München, D-80290 München, Germany (devries@ma.tum.de). This author's research was supported in part by the German Federal Department of Education, Science, Research, and Technology grant GR7TM1/02.05.

FIG. 2.1. Picture of a simple $(8,3)$ -antiweb.

a *rim-edge*. We define the following *distance function* for the vertices of an antiweb with $i > j$ by

$$\text{dist}(v_i, v_j) := \min(i - j, n + j - i).$$

We denote this antiweb by $\mathcal{AW}(v_1, v_2, \dots, v_n)$. An example of an $(8,3)$ -antiweb is drawn in Figure 2.1. The class of (\cdot, t) -antiwebs is referred to as (t) -antiwebs. Thus (2) -antiwebs are odd cycles. Our definition of an antiweb is slightly different from the one given by Trotter [18] because our definition includes odd cliques (for $n = 2t - 1$). An (n, t) -antiweb contains n different t -cliques, namely, each t -clique \mathcal{T}_i on the vertices $\{v_i, v_{i+1}, \dots, v_{i+t-1}\}$ for $i = 1, 2, \dots, n$ (where indices $j > n$ are reduced to $1 + ((j - 1) \bmod n)$). We refer to $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n$ as the *generators* of the antiweb. The inequality $\sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$ is the *antiweb inequality* described in [18]. For $n = 2t - 1$ it is a clique inequality and for $t = 2$ it is a cycle inequality. Euler, Jünger, and Reinelt [12] generalized inequalities of some antiwebs (the class of odd anticycles) to *independence system polytopes*; Laurent [14] studied the full class of antiweb inequalities for independence system polytopes. Schulz [16] and Müller and Schulz [15] generalized antiwebs further to the general setting of *transitive packing*.

We write the union of two sets V, W as $V \dot{\cup} W$ if we want to emphasize that $V \cap W = \emptyset$.

The stable set problem for antiwebs is solvable in polynomial time, as they belong to the class of circular arc graphs for which Golumbic and Hammer [13] proved that the stable set problem is solvable in polynomial time. A description by inequalities is given by Dahl [10] for the stable set polytopes of 3-antiwebs, but neither the question of their separation is treated nor is anything said about (t) -antiweb polytopes for $t > 3$. Efficient separation algorithms (that is, algorithms for the following problem: given an x , find a violated inequality in a given class or conclude that none exists) for generalizations of (t) -antiweb inequalities are given in [8, 6, 11]. Furthermore, in [7, 6, 11] it is shown that separation of antiwebs is already NP-hard.

3. General applicable subdivision operations. In this section we study three different procedures to obtain new facets of a graph from facets of a smaller, related graph. The first two procedures are known from the literature and therefore they are reviewed quickly. The third one, although described by Barahona and Mahjoub [1], is applied under new circumstances. For these procedures, we will consider conditions that preserve only validity as well as conditions that preserve additionally being a facet.

The operation of adding an apex and its polyhedral consequences are well known; see [9]. It turns out that adding an apex is the same as substituting the graph into one vertex of a K_2 .

DEFINITION 3.1 (apex and spokes). *Given a graph $G = (V, E)$ and a vertex $v \notin V$, we define the graph G^v on the vertex set $V \cup \{v\}$ and edge set $E \cup \{\{u, v\} : u \in V\}$. The vertex v is called the apex of G^v . The edges of the form $\{v_{0_i}, v\}$ with $v \in V$ of a graph $G^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$ are called spokes. If G is an antiweb, then the edges of G in $G^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$ are again referred to as cross-edges, and their type is the type that they had in the antiweb.*

REMARK 3.2. *Henceforth, whenever we speak of G^v we will assume that $v \notin V(G)$.*

Our objective is to find facet-inducing inequalities whose support are graphs that we call “antiweb-wheels.” An example is given in Figure 4.2. Our starting point will be the facet-inducing antiweb inequality for an antiweb. (See Figure 2.1.) The next proposition completes the first step in obtaining a facet-inducing inequality for the graph in Figure 4.1.

PROPOSITION 3.3. *Given a graph G and an inequality $a^\top x \leq \alpha$ which defines a facet of $STAB(G)$ with $\alpha > 0$, then the inequality $a^\top x + \alpha x_v \leq \alpha$ defines a facet of $STAB(G^v)$.*

Proof. The validity is immediate. From the facetness for $STAB(G)$ we know that a $|V| \times |V|$ matrix Y exists so that its column vectors belong to $STAB(G)$ and span the old facet. Since $\alpha > 0$ the columns of Y are linearly independent, so Y is nonsingular. Furthermore, all column vectors from Y belong also to the new facet of $STAB(G^v)$. Another column vector which belongs to both is the vector χ^v . Let

$$Y^{\text{new}} = \begin{pmatrix} & 0 \\ Y & \vdots \\ & 0 \\ 0 \dots 0 & 1 \end{pmatrix}.$$

Clearly, Y^{new} has full rank. \square

The operation of adding an apex permits not only the transition from the graph in Figure 2.1 to the one in Figure 4.1 but also from Figure 4.1 to the one in Figure 4.6.

As our objective is to construct facets for the graph in Figure 4.2 from facets of the graph in Figure 2.1, we need two more graph operations. One of them is double edge subdivision. The following result by Wolsey [19] shows how a valid inequality is transformed by double edge subdivision into another valid inequality.

PROPOSITION 3.4 (double edge subdivision). *Let $G = (V, E)$ be a graph and $a^\top x \leq \alpha$ ($a \geq 0, \alpha > 0$) be valid for $STAB(G)$ with $\{p, q\} \in E$ and $\gamma = \min\{a_p, a_q\}$. Let G' be the graph obtained from G by replacing $\{p, q\}$ with the path $p - u - v - q$, where $u, v \notin V$. Then $a^\top x + \gamma x_u + \gamma x_v \leq \alpha + \gamma$ is valid for $STAB(G')$.*

Proof. Consider a stable set S' of G' and its incidence vector $\chi^{S'}$. If $p, q \in S'$, then $u, v \notin S'$. Without loss of generality, we can assume $a_p = \gamma$. Notice that $S = S' - p$ is a stable set of G . Plugging χ^S into the valid inequality of $STAB(G)$ shows $a^\top \chi^S \leq \alpha$ and then $a^\top \chi^{S'} + \gamma \chi_u^{S'} + \gamma \chi_v^{S'} = a^\top \chi^S + \gamma \leq \alpha + \gamma$. If at most one of p, q is in S' , then at most one of u, v is in S' . Again, $S = S' \setminus \{u, v\}$ is stable in G , and validity of the initial inequality of $STAB(G)$ yields $a^\top \chi^{S'} + \gamma \chi_u^{S'} + \gamma \chi_v^{S'} \leq a^\top \chi^S + \gamma \leq \alpha + \gamma$. \square

The set of neighbors of a vertex v of G is denoted by $N(v) := \{w : \{v, w\} \in E\}$.

For the facetness version of Proposition 3.4, we use a special form of a result by Wolsey [19]; for a proof of the forward direction, see [3, Lem. 2.3.6].

PROPOSITION 3.5. *Let $G = (V, E)$ be a graph and $a^\top x \leq \alpha$ ($a \geq 0, \alpha > 0$) be facet-inducing for $STAB(G)$ with $\{p, q\} \in E$ and $a_p \geq a_q = \gamma$.*

1. *There exists a stable set S in G with $a^\top \chi^S = \alpha$ and $p, q \notin S$ and*

2. there exists a stable set S in G with $a^\top \chi^S = \alpha$, $p \in S$, $q \notin S$ and $h \notin S$ for all $h \in N(q) \setminus \{p\}$

if and only if $a^\top x + \gamma x_u + \gamma x_v \leq \alpha + \gamma$ is facet-inducing for $STAB(G')$.

Proof. For the reverse direction, assume $I' : a^\top x + \gamma x_u + \gamma x_v \leq \alpha + \gamma$ is facet-inducing for $STAB(G')$. Then there exists a stable set S of G with $p, u \notin S$ such that its incidence vector satisfies I' with equality. Therefore, $v \in S$ and $q \notin S$. Hence $S' = S \setminus \{v\}$ is a stable set of G with $p, q \notin S'$ that satisfies $I : a^\top x \leq \alpha$ with equality. There also must exist a stable set S of G with $u, v \notin S$ whose incidence vector satisfies I' with equality. Hence $p, q \in S$ and all the neighbors of q in G' do not belong to S . Thus $S \setminus \{q\}$ satisfies condition 2. \square

In this paper, we often use $\gamma = 1$ when we utilize Proposition 3.5. We note that if $a^\top x \leq \alpha$ is not an edge inequality, then condition 1 of Proposition 3.5 is automatically satisfied. Furthermore, if $\deg(q) = 2$, then condition 2 in Proposition 3.5 is also automatically satisfied. For the reverse operation of replacing a path of length 3 with a path of length 1 we need the following result from [1, Thm. 2.5]. Part 1 gives a condition under which validity of the inequality carries over, while part 2 gives a condition that preserves facetness.

PROPOSITION 3.6. *Let $G = (V, E)$ be a graph. Let $a^\top x \leq \alpha$ be a valid inequality of $STAB(G)$. Suppose that G contains a path $p - u - v - q$ such that u and v are of degree 2 and $a_u = a_v$. Set $\beta = a_u$. Let $G' = (V', E')$ be the graph obtained from G by replacing the path by the edge $\{p, q\}$. Let $\bar{a}_u = a_u$ for $u \in V'$ and $\bar{\alpha} = \alpha - \beta$.*

1. *If $\beta \leq \min\{a_p, a_q\}$, then $\bar{a}^\top x \leq \bar{\alpha}$ is valid for $STAB(G')$.*
2. *If, furthermore, $a_p = \beta$ and the original inequality induces a facet, then $\bar{a}^\top x \leq \bar{\alpha}$ is facet-inducing for $STAB(G')$.*

We now give a new theorem for lifting valid inequalities and facets from a graph to another graph where all edges incident with a single vertex are subdivided once. This graph operation was already introduced by Barahona and Mahjoub [1, Thm. 2.3]. Our result differs from theirs in that we relax one prerequisite of their theorem while we replace their other condition by a stronger requirement. Our resulting theorem cannot be proved with the generalizing parameter p as in [1, Thm. 2.3]. A simple calculation demonstrates that if the incidence-structure for facetness of Theorem 3.7 has to be maintained and the inequality should be valid, then the generalizing parameter p of Theorem [1, Thm. 2.3] can only have value 1 for our theorem.

THEOREM 3.7 (star-subdivision). *Let $G = (V, E)$ be a graph and $a^\top x \leq \alpha$ be a valid inequality. Let v be a vertex of G and $N = \{v_1, \dots, v_{k+1}\}$ be the set of neighbors of v , where $k \geq 1$. Let $G' = (V', E')$ be the graph obtained from G by subdividing each edge $\{v, v_i\}$ with a new vertex v'_i for $i = 1, \dots, k + 1$. Set*

$$\bar{a}_u = a_u \text{ for } u \in V \setminus \{v\},$$

$$\bar{a}_v = k,$$

$$\bar{a}_{v'_i} = 1 \text{ for } i = 1, 2, \dots, k + 1,$$

$$\bar{\alpha} = \alpha + k.$$

1. *Suppose that $a_v = 1$. Then $\bar{a}^\top x \leq \bar{\alpha}$ is a valid inequality of $STAB(G')$.*
2. *If, additionally, $a^\top x \leq \alpha$ defines a nontrivial facet of $STAB(G)$ and for each $i = 1, \dots, k+1$ there exists a stable set \tilde{S}_i such that $a^\top \chi^{\tilde{S}_i} = \alpha$ and $\tilde{S}_i \cap N = \{v_i\}$, then $\bar{a}^\top x \leq \bar{\alpha}$ defines a facet of $STAB(G')$.*

Proof. For the first part suppose S' is a stable set that violates the new inequality:

1. If $v \in S'$ and the left-hand side $> \alpha + k$, then v'_i is not in S' for all i . So $S' \setminus \{v\}$ is a stable set in G and violates the old inequality because the left-hand side would

- be $> \alpha$.
2. If $v \notin S'$, v'_i in S' for all i and the left-hand side $> \alpha + k$, then each v_i is not in S' . So $(S' \setminus \{v'_1, v'_2, \dots, v'_{k+1}\}) \cup \{v\}$ is a stable set for the old graph. The left-hand side is $> \alpha + k - (k + 1) + 1 = \alpha$.
 3. If $v \notin S'$, not all v'_i in S' and the left-hand side $> \alpha + k$. Let U be the set of v'_i in S' . Then $|U| \leq k$ and $S' \setminus U$ is a stable set for the old graph. The left-hand side is $> \alpha + k - |U| \geq \alpha$.

The proof of the second part is the same as that given by Barahona and Mahjoub [1], except that they require that p and $k + 1$ are coprime to ensure that a $(k + 1 - p)$ -circulant $(k + 1) \times (k + 1)$ -matrix has full rank, whereas we utilize that a k -circulant $(k + 1) \times (k + 1)$ -matrix has full rank. \square

REMARK 3.8. *Actually only $a_v > 0$ is necessary in Theorem 3.7, but in this case the definition of \bar{a} is more complicated.*

From now on, the terms *facetly double subdivision* and *facetly star-subdivision* will be used if the subdivision satisfies the conditions required to transform a facet into a facet.

So we have assembled the tools to get from the graph in Figure 4.1 to the graph in Figure 4.2. First we apply star-subdivision at the vertices v_1, v_6, v_8 ; see Figure 4.4. Next we doubly subdivide the edges $\{v_6, z\}, \{v_0, v_7\}, \{v_2, v_4\}$. For $\{v_3, v_5\}$, we will doubly subdivide it to the path $v_3 - u - v - v_5$ and then doubly subdivide $\{u, v\}$. The result is shown in Figure 4.2. Additionally, if we would replace the path of length three from v_6 to v_8 by the edge $\{v_6, v_8\}$, then we obtain the graph in Figure 4.5.

Ultimately, we want to obtain facets for graphs like the one in Figure 4.2 from the antiweb facet of the graph in Figure 4.1. We have already seen a series of results addressing this issue. However, in order to apply them, we need to check certain conditions in every step. In the next section, we will show various results of the type “if operations A and B are applicable to a graph, then we can first perform A and afterwards B is still applicable.” These results allow us then to check applicability only on the initial graph. Next we state and prove some results related to the star-subdivision that will be useful in later sections.

LEMMA 3.9 (incidence vectors of star subdivided faces). *A stable set S whose incidence vector satisfies with equality the valid inequality $\bar{a}^\top x \leq \bar{\alpha}$ of Theorem 3.7 (constructed from a valid inequality $a^\top x \leq \alpha$ of the underlying graph before the star-subdivision was performed) fulfills one of the following conditions:*

1. $v \in S$,
2. $v \notin S$, but all $v'_i \in S$, or
3. $v \notin S$, and exactly k of the v'_i belong to S .

Proof. Suppose for a contradiction that we have a stable set S with $v \notin S$ and at most $k - 1$ of the v'_i belong to S . Then $S \setminus \{v'_1, v'_2, \dots, v'_{k+1}\}$ is a stable set of the unsubdivided underlying graph, but it violates that corresponding inequality. \square

LEMMA 3.10 (converse of Theorem 3.7). *The converse of Theorem 3.7, part 2 is true. That is, if one of the sets \tilde{S}_i does not exist, then the resulting face is not a facet.*

Proof. So assume that there is no set \tilde{S} with (say) $\tilde{S} \cap N = \{v_1\}$. Then we claim that the new face $\bar{a}^\top x \leq \bar{\alpha}$ is contained in the face $x_v + x_{v'_1} \leq 1$. For this we will show that $\bar{a}^\top x = \bar{\alpha}$ implies $x_v + x_{v'_1} = 1$.

Considering the possible types for a stable set S' with $\bar{a}^\top \chi^{S'} = \alpha$ characterized in Lemma 3.9, we notice that types 1 and 2 obviously fulfill $x_v + x_{v'_1} = 1$. So we need to consider stable sets S' of type 3. A set S' of type 3 could contradict the equation

$x_v + x_{v'_1} = 1$ only if v'_1 does not belong to S' . If, additionally, $v_1 \notin S'$, then the stable set $S' \cup \{v'_1\}$ would violate $\bar{a}^\top x \leq \bar{\alpha}$ (because $\bar{a}_{v'_1} = 1 > 0$).

So we can assume $v_1 \in S'$. If $S' \cap N$ were equal to $\{v_1\}$, then the set S' without the vertices of type v'_i would belong to the facet $a^\top x \leq \alpha$ and would intersect N only in $\{v_1\}$, contrary to the assumption that every stable set from the facet which contains v_1 contains another neighbor of v .

However, by Lemma 3.9 the vector $\chi^{S'}$ cannot lie in the facet, if $v \in S'$ and $|S' \cap N| < k$, thereby providing a contradiction. \square

LEMMA 3.11 (necessity of condition in Theorem 3.7). *If the valid inequality (with $a_v = 1$) to start with in Theorem 3.7 is not a facet, then the resulting face is not a facet.*

Proof. So assume the face $a^\top x \leq \alpha$ of G is contained in another face $b^\top x \leq \beta$. As $a_v = 1$ we can choose for $b^\top x \leq \beta$ an inequality with $b_v = 1$, because the inequality $a^\top x \leq \alpha$ can then be described as the convex combination of other valid inequalities; as $a_v = 1$ one of them has to have the nonzero coefficient for x_v . Then we want to show that the face $\bar{a}^\top x \leq \bar{\alpha}$ is contained in $\bar{b}^\top x \leq \bar{\beta}$. For this it suffices to show for every stable set S' of G' that $\bar{a}^\top \chi^{S'} = \bar{\alpha}$ implies $\bar{b}^\top \chi^{S'} = \bar{\beta}$.

Again we use the characterization of stable sets S' with $\bar{a}^\top \chi^{S'} = \alpha$ by Lemma 3.9. For these cases we obtain the following:

1. $a^\top \chi^{S'-v} = \alpha$ holds; hence follows $b^\top \chi^{S'-v} = \beta$ and $\bar{b}^\top \chi^{S'} = \bar{\beta}$.
2. $a^\top \chi^{S'+v-N'} = \alpha$ holds; hence follows $b^\top \chi^{S'+v-N'} = \beta$ and $\bar{b}^\top \chi^{S'} = \bar{\beta}$.
3. So we can assume $|S' \cap N'| = k$. Then $a^\top \chi^{S'-N'} = \alpha$; hence $b^\top \chi^{S'-N'} = \beta$; hence $\bar{b}^\top \chi^{S'} = \bar{\beta}$. \square

4. Interaction of the subdivision operations. In this section we will study the interaction of the three subdivision operations from the preceding section. However, first we define a class of graphs that can be generated from antiwebs by repeated application of the three subdivision operations; then we give several examples.

DEFINITION 4.1 (antiweb- s -wheel). *Let $s \geq 0$. Given an (n, t) -antiweb $G_1 = (V_1, E_1)$ with $n \not\equiv 0 \pmod t$ and an arbitrary partition \mathcal{E}, \mathcal{O} of $V_1 = \{1, 2, \dots, n\}$. Consider a subdivision G of $G_1^{v_{0_1}, v_{0_2}, \dots, v_{0_s}}$. Let $P_{0_i, j}$ denote the path obtained from subdividing the edge $\{v_{0_i}, v_j\}$ and let $P_{i, j}$ (for v_i, v_j adjacent in G_1) denote the path obtained from subdividing the edge $\{v_i, v_j\}$. This graph G is a simple antiweb- s -wheel if the following four conditions are fulfilled:*

1. For all $1 \leq i \leq s$ is the length of $P_{0_i, j}$ even for $j \in \mathcal{E}$ and odd for $j \in \mathcal{O}$.
2. The length of the path $P_{i, j}$ is even for $i \in \mathcal{E}$ and $j \in \mathcal{O}$ or $j \in \mathcal{E}$ and $i \in \mathcal{O}$.
3. The length of the path $P_{i, j}$ is odd for $i, j \in \mathcal{O}$.
4. The length of the path $P_{i, j}$ is odd for $i, j \in \mathcal{E}$.

A simple antiweb- s -wheel is called *proper* if $P_{i, j}$ is of length at least 2 for all paths with at least one end in \mathcal{E} . A proper antiweb- s -wheel is called *basic* with respect to a given partition \mathcal{E}, \mathcal{O} if all involved paths have minimal length. See Figure 4.1 for a simple basic antiweb-1-wheel. Figure 4.2 depicts a simple proper antiweb-1-wheel with nontrivial partition $\mathcal{E} \cup \mathcal{O}$ obtained from Figure 4.1 via subdivisions (Figures 4.3 and 4.4) and finally double edge subdivisions. Figure 4.5 depicts a simple nonproper antiweb-1-wheel with nontrivial partition $\mathcal{E} \cup \mathcal{O}$. Finally, Figure 4.6 displays a simple $(8, 3)$ -antiweb-2-wheel. For $t = 2$, antiweb-wheels are the 1-wheel defined in [5].

Next we want to study the interaction of the different operations with respect to facetness. Our goal is to establish new facets for antiweb- s -wheels.

LEMMA 4.2 (iterative application of Theorem 3.7). *Assume that each of two vertices v, w of a graph G fulfill the conditions of Theorem 3.7 with respect to G and*

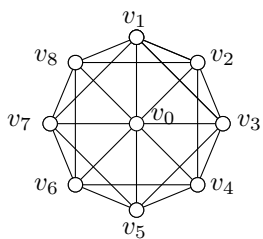


FIG. 4.1. Picture of a simple basic $(8,3)$ -antiweb-1-wheel, where no edge is subdivided and all vertices (on the rim) belong to \mathcal{O} .

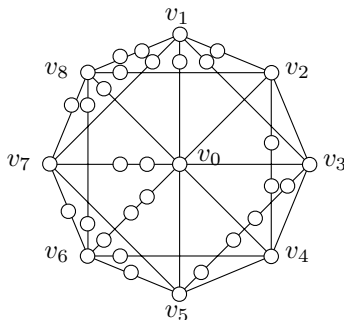


FIG. 4.2. Picture of a simple proper $(8,3)$ -antiweb-1-wheel with $\mathcal{E} = \{v_1, v_6, v_8\}$ and $\mathcal{O} = \{v_2, v_3, v_4, v_5, v_7\}$.

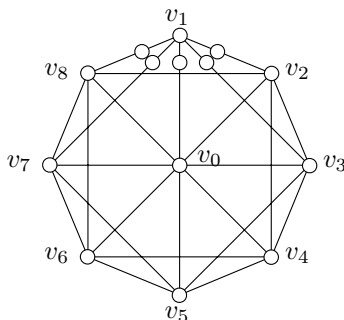


FIG. 4.3. Picture of a simple proper $(8,3)$ -antiweb-1-wheel with $\mathcal{E} = \{v_1\}$ and $\mathcal{O} = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

the facet-inducing inequality $a^\top x \leq \alpha$. Let G' be the graph constructed from v as in Theorem 3.7. Then w fulfills the conditions of Theorem 3.7 with respect to G' and $\bar{a}^\top x \leq \bar{\alpha}$.

Proof. Let the neighbors of v in G be $\{v_1, v_2, \dots, v_{k+1}\}$ and the neighbors of w in G be $\{w_1, w_2, \dots, w_{l+1}\}$. Let $\tilde{S}_1, \tilde{S}_2, \dots, \tilde{S}_{k+1}$ and $\tilde{U}_1, \tilde{U}_2, \dots, \tilde{U}_{l+1}$ be the stable sets in the facet with $\tilde{S}_i \cap \{v_1, v_2, \dots, v_{k+1}\} = v_i$ and $\tilde{U}_i \cap \{w_1, w_2, \dots, w_{l+1}\} = w_i$.

There are two cases, depending on the adjacency of v and w .

1. v and w are nonadjacent in G . Let $N = \{v_1, v_2, \dots, v_{k+1}\}$ be the set of neighbors of v in G and $M = \{w_1, w_2, \dots, w_{l+1}\}$ be the set of neighbors of w in G . Notice that $M' = M$ is, additionally, the set of neighbors of w in G' , as v, w are nonadjacent.

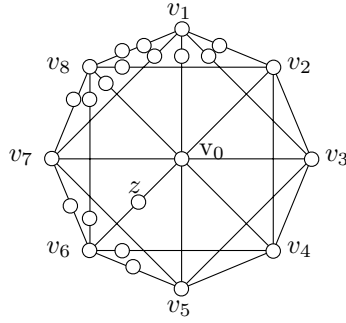


FIG. 4.4. Picture of a simple proper (8,3)-antiweb-1-wheel with $\mathcal{E} = \{v_1, v_6, v_8\}$ and $\mathcal{O} = \{v_2, v_3, v_4, v_5, v_7\}$ without facetly edge subdivision performed.

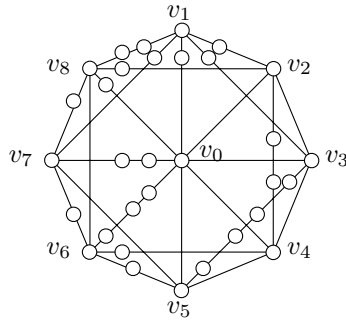


FIG. 4.5. Picture of a simple, nonproper (8,3)-antiweb-1-wheel with $\mathcal{E} = \{v_1, v_6, v_8\}$ and $\mathcal{O} = \{v_2, v_3, v_4, v_5, v_7\}$.

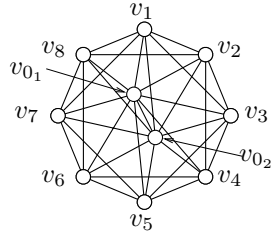


FIG. 4.6. A simple (8,3)-antiweb-2-wheel, where no edge is subdivided and all vertices on the rim belong to \mathcal{O} .

For $i = 1, 2, \dots, l + 1$ let

$$\tilde{U}'_i = \begin{cases} \tilde{U}_i \setminus \{v\} \cup \{v'_1, v'_2, \dots, v'_{k+1}\} & \text{if } v \in \tilde{U}_i, \\ \tilde{U}_i \cup \{v\} & \text{if } v \notin \tilde{U}_i. \end{cases}$$

Notice that $a^\top \chi^{\tilde{U}_i} = \alpha$ implies $\bar{a}^\top \chi^{\tilde{U}'_i} = \bar{\alpha}$. Now we want to show that \tilde{U}'_i fulfills the conditions of Theorem 3.7 with respect to $\bar{a}^\top x \leq \bar{\alpha}$ and G' . Notice first that $\bar{a}_w = a_w$ and $a_w = 1$ by assumption. Then consider $\tilde{U}'_i \cap M'$; by construction $w_i \in \tilde{U}_i$ and by assumption of this case $w_i \neq v$, hence $w_i \in \tilde{U}'_i$ and $w_i \in \tilde{U}'_i \cap M'$ follows. Now assume that $w_j \in \tilde{U}'_i$. If $w_j \in \tilde{U}_i$, then $j = i$; otherwise, either $w_j = v$ which is impossible or $w_j \in \{v'_1, v'_2, \dots, v'_{k+1}\}$ which is also impossible. Hence we succeeded in proving $\tilde{U}'_i \cap M' = \{w_i\}$, and all conditions of Theorem 3.7 are satisfied.

2. v and w are adjacent in G . Without loss of generality, assume that $v_1 = w$ and $w_1 = v$. Let $N = \{v_1, v_2, \dots, v_{k+1}\}$ be the set of neighbors of v in G and $M = \{w_1, w_2, \dots, w_{l+1}\}$ be the set of neighbors of w in G . Notice that $M' = \{v'_1, w_2, w_3, \dots, w_{l+1}\}$, the set of neighbors of w in G' , is different from M . For $i = 1, 2, \dots, l + 1$ let

$$\tilde{U}'_i = \begin{cases} \tilde{U}_i \setminus \{v\} \cup \{v'_1, v'_2, \dots, v'_{k+1}\} & \text{if } v \in \tilde{U}_i, \\ \tilde{U}_i \cup \{v\} & \text{if } v \notin \tilde{U}_i. \end{cases}$$

Notice that $a^\top \chi^{\tilde{U}_i} = \alpha$ implies $\bar{a}^\top \chi^{\tilde{U}'_i} = \bar{\alpha}$. Now we want to show that \tilde{U}'_i fulfills the conditions of Theorem 3.7 with respect to $\bar{a}^\top x \leq \bar{\alpha}$ and G' . Notice first that $\bar{a}_w = a_w$ and $a_w = 1$ by assumption. As $\tilde{U}'_i = \tilde{U}_i \cup \{v\}$ for $i = 2, 3, \dots, l + 1$ it follows that $\tilde{U}'_i \cap M' = \{w_i\}$ for $i = 2, 3, \dots, l + 1$. For $i = 1$ holds $\tilde{U}'_1 = \tilde{U}_1 \setminus \{v\} \cup \{v'_1, v'_2, \dots, v'_{k+1}\}$ and thereby $\tilde{U}'_1 \cap M' = \{v'_1\}$ and all conditions of Theorem 3.7 are satisfied. \square

LEMMA 4.3. *Let G be a graph. Suppose G' is obtained from G by replacing an edge $\{p, q\}$ by $p - u - v - q$. Suppose $a^\top x \leq \alpha$ is not an edge inequality. If $a^\top x \leq \alpha$ is facet-inducing for $STAB(G)$ with $a \geq 0$, $a_p \geq a_q = \gamma$, and $\alpha > 0$, and $\{p, q\}$ satisfies condition 2 of Proposition 3.5 with respect to $I : a^\top x \leq \alpha$, then $I' : a^\top x + \gamma x_u + \gamma x_v \leq \alpha + \gamma$ is facet-inducing for $STAB(G')$. Moreover, $\{p, u\}$, $\{u, v\}$, and $\{v, q\}$ satisfy condition 2 of Proposition 3.5 with respect to I' ; if $\{p_1, q_1\} \in E(G) \cap E(G')$ satisfies condition 2 of Proposition 3.5 with respect to I , then $\{p_1, q_1\}$ satisfies condition 2 of Proposition 3.5 with respect to I' . In addition, if $\{p_1, q_1\} \in E(G) \cap E(G')$ satisfies condition 2 of Proposition 3.5 with respect to I' , then $\{p_1, q_1\}$ satisfies condition 2 of Proposition 3.5 with respect to I unless $a_p > a_q$ and $q = p_1$.*

Proof. First we prove the forward implication. It follows from Proposition 3.5 that I' is facet-inducing for $STAB(G')$. Clearly, any edge of the path $p - u - v - q$ satisfies the hypotheses in Proposition 3.5 because $\deg(u) = \deg(v) = 2$. Now suppose $\{p_1, q_1\} \in E(G) \cap E(G')$. As $\{p_1, q_1\}$ (with respect to G) satisfies the hypotheses, there is a stable set of U whose incidence vector χ^U satisfies $a^\top x \leq \alpha$ with equality and $p_1 \in U$, $q_1 \notin U$, and $f \notin U$ for all $f \in N(q_1) \setminus \{p_1\}$. If $q_1 \notin \{p, q\}$, then we can extend U to U' by setting $U' = U \cup \{u\}$ if $p \notin U$, or by setting $U' = U \cup \{v\}$ if $q \notin U$. Now U' does the job. If $q_1 = p$, then we can extend U to U' by setting $U' = U \cup \{v\}$ (as $q \notin U$); therefore U' does the job. If $q_1 = q$, then we can extend U to U' by setting $U' = U \cup \{u\}$ (as $p \notin U$); therefore U' does the job.

Now we do the reverse implication. Suppose (p_1, q_1) satisfy condition 2 of Proposition 3.5 with respect to I' . Let S be a stable set that fulfills the condition. Assume $a_{p_1} \geq a_{q_1} = \beta$. We consider several cases:

$p = q_1$: Then $q \neq p_1$, $p_1 \in S$, and $q_1, u \notin S$. Because the incidence vector of S satisfies I' with equality, we may assume $v \in S$. Hence $S \setminus \{v\}$ satisfies condition 2 of Proposition 3.5 with respect to I .

$q = q_1$: Then $p \neq p_1$, $p_1 \in S$, and $q_1, v \notin S$. Because the incidence vector of S satisfies I' with equality, we may assume $u \in S$. Hence $S \setminus \{u\}$ satisfies condition 2 of Proposition 3.5 with respect to I .

$q_1 \neq p$ and $q_1 \neq q$: (However, one of p, q may be p_1 .) We have to consider several subcases, namely, (a) $p, q \in S$ and $u, v \notin S$, (b) $p, v \in S$ and $u, q \notin S$ and (c) $q, u \in S$ and $p, v \notin S$. It is easy to see that in each of these cases a feasible set for condition 2 is at hand. As S satisfies I' with equality, the only remaining cases are (a) $u \in S$ and $p, q, v \notin S$ and (b) $v \in S$ and $p, q, u \notin S$. Then $S \setminus \{u\}$ and $S \setminus \{v\}$ fulfill condition 2 in their respective cases. \square

LEMMA 4.4 (Theorem 3.7 and Proposition 3.5 commute). *Assume that the vertex v of a graph G fulfills the conditions of Theorem 3.7 with respect to G and the facet-inducing inequality $a^\top x \leq \alpha$. Furthermore, assume that the edge $\{u, w\}$ fulfills the requirements of Proposition 3.5 with the particular set S and $a_u \geq a_w$ and that $a^\top x \leq \alpha$ is not the edge inequality $x_u + x_w \leq 1$.*

1. *Let G' be the graph constructed from v as in Theorem 3.7. Then G' fulfills the conditions of Proposition 3.5 for $\{u, w\}$ if u, w are adjacent in G' or otherwise for one of the edges $\{u, w'\}$ or $\{u', w\}$ with respect to $\bar{a}^\top x \leq \bar{\alpha}$. Here u' and w' denote the new vertices created on the edges $\{v, u\}$ and $\{v, w\}$ of G .*
2. *Let G'' be the graph constructed from $\{u, w\}$ as in Proposition 3.5 and $\bar{a}^\top x \leq \bar{\alpha}$ the corresponding inequality. Then G'' fulfills the conditions of Theorem 3.7 for v with respect to $\bar{a}^\top x \leq \bar{\alpha}$.*

Proof. For the first part we have to distinguish three cases:

1. $u, w \neq v$. First notice that $\bar{a}_u = a_u \geq a_w = \bar{a}_w$. If $w \in N$, then $v \notin S$ and thereby $S \cup \{v\}$ does the job. If $w \notin N$, then depending on $v \in S$ or $v \notin S$ either $(S \setminus \{v\}) \cup \{v'_1, v'_2, \dots, v'_{k+1}\}$ or $S \cup \{v\}$ does the job (as nothing in the neighborhood of w is changed).
2. $w = v$ (and $u \in N$). Say $u = v_i$. Let $w_{\text{new}} = v'_i$. Now notice that $\deg(w_{\text{new}}) = 2$. With the remark after Proposition 3.5 this shows that the conditions of Proposition 3.5 are satisfied.
3. $u = v$ (and $w \in N$). Say $w = v_i$. Let $u_{\text{new}} = v'_i$. Notice that $1 = \bar{a}_{u_{\text{new}}} = a_u \geq a_w = \bar{a}_w$. Because of $u = v$ it follows that $v \in S$. Now $(S \setminus \{v\}) \cup \{v'_1, v'_2, \dots, v'_{k+1}\}$ does the job.

For the second part, notice first that the subdivision of $\{u, w\}$ with y, z (thereby creating the path $u - y - z - w$) cannot change the coefficient of v from 1 to anything wrong. So validity is already guaranteed. If u, w are not adjacent to v , then the sets \hat{S}_i need to be augmented only with y or z to make them incident with the facet of $\bar{a}^\top x \leq \bar{\alpha}$. If u and/or w are adjacent to v , then still every set can be augmented with either y or z to keep it in the face $\bar{a}^\top x \leq \bar{\alpha}$. If finally u or w is v (say $u = v$), then the sets \hat{S}_i that do not contain w can be augmented with z without disturbing the neighborhood of v while maintaining incidence with $\bar{a}^\top x \leq \bar{\alpha}$. For sets \hat{S}_i that do contain w we can augment with y , and the new set intersects the neighborhood of v only in y . \square

Next we show that a subdivision procedure is (facetly) applicable to G^v if it is (facetly) applicable to G .

LEMMA 4.5 (lifting of Theorem 3.7). *Let $G = (V, E)$ be a graph and $a^\top x \leq \alpha$ be a nontrivial valid inequality. Let v be a vertex of G and $N = \{v_1, \dots, v_{k+1}\}$ be the neighbor set of v , where $k \geq 1$. Let $G' = G^{v_{k+2}}$. Extend a to G' according to Proposition 3.3 by $a'_u = a_u$ for all $u \in V$ and $a'_{v_{k+2}} = \alpha$.*

1. *If $(G, v, a^\top x \leq \alpha)$ fulfills the conditions of Theorem 3.7, part 1, then $(G', v, a'^\top x \leq \alpha)$ fulfills these conditions again.*
2. *If $(G, v, a^\top x \leq \alpha)$ additionally fulfills the conditions of Theorem 3.7, part 2, then $(G', v, a'^\top x \leq \alpha)$ fulfills these conditions again.*

Furthermore, if $(G^{v_{k+2}}, v, a'^\top x \leq \alpha)$ with $a'_{v_{k+2}} = \alpha$ fulfills the conditions of Theorem 3.7, part 1 and/or part 2 and $a'^\top x \leq \alpha$ is not a 3-cycle inequality, then $(G, v, a^\top x \leq \alpha)$ fulfills them again.

Proof. Validity and facetness of $a'^\top x \leq \alpha$ follow directly from Proposition 3.3. For part 1 there is nothing to show. For part 2 we know that for $i = 1, \dots, k + 1$,

there exists a stable set \tilde{S}_i such that $a^\top \chi^{\tilde{S}_i} = \alpha$ and $\tilde{S}_i \cap N = \{v_i\}$. We set $\tilde{S}'_i = \tilde{S}_i$ for $i = 1, \dots, k+1$ and $\tilde{S}'_{k+2} = \{v_{k+2}\}$. Obviously, the conditions of part 2 are fulfilled for $i = 1, \dots, k+1$. For $i = k+2$ the facts $a'^\top \chi^{\tilde{S}'_{k+2}} = \alpha$ and $\tilde{S}'_{k+2} \cap N = \{v_{k+2}\}$ are clear.

For the other direction, note that part 1 is easy, while for part 2 it suffices that all the \tilde{S}'_i carry over from G' to G (except the one corresponding to v_{k+2}). \square

LEMMA 4.6 (lifting of Proposition 3.5). *Let $G = (V, E)$ be a graph, $a^\top x \leq \alpha$ be a nontrivial facet-inducing inequality, and $\{p, q\}$ be an edge of G . If condition 2 of Proposition 3.5 is fulfilled for $(G, a^\top x \leq \alpha, p, q)$, then conditions 1 and 2 of Proposition 3.5 are also fulfilled for the graph $G' = G^w$, the inequality $a^\top x = a'^\top x + \alpha x_w \leq \alpha$, and the vertices p, q . Furthermore, if $(G^w, p, q, a'^\top x \leq \alpha)$ with $a_w = \alpha$ fulfills the conditions 1 and 2 of Proposition 3.5 and if $a'^\top x \leq \alpha$ is not a 3-cycle inequality, then $(G, p, q, a^\top x \leq \alpha)$ fulfills them.*

Proof. Validity and facetness of $a'^\top x \leq \alpha$ follow by Proposition 3.3 from these properties for $a^\top x \leq \alpha$. Consider the stable set $S' = \{w\}$. Notice $a'^\top \chi^{S'} = \alpha$ and $\chi_p^{S'} = \chi_q^{S'} = 0$. So condition 1 of Proposition 3.5 is satisfied. Condition 2 is verified for G' with the same set which fulfills it for G .

For the converse direction notice that the two required sets directly carry over from G' to G . \square

LEMMA 4.7 (subdivision of the spokes). *Let $G = (V, E)$ be a graph, $a^\top x \leq \alpha$ be a facet-inducing inequality, and q be a vertex of G so that $a^\top x \leq \alpha$ is not the inequality $x_q \leq 1$. Then conditions 1 and 2 of Proposition 3.5 are fulfilled for the graph G^q together with the inequality $a'^\top x = ax + \alpha x_p \leq \alpha$ and the edge $\{p, q\}$.*

Proof. Validity and facetness are easy again. As $a'_p \geq a'_q$ the stable set $S = \{p\}$ does the job. Furthermore, the new inequality is not an edge inequality, because the old inequality was not the constraint $x_q \leq 1$. \square

5. Facet-inducing antiweb-wheels. In section 3, we considered three operations that turn an antiweb into an antiweb-s-wheel. However, to ensure that each operation on the inequalities preserves facetness rather than only validity, we need to check various conditions. This could be cumbersome. To ease this, we proved in section 4 commutative and successive properties of these facetly operations. With these properties, we need only to check the various conditions on the initial graph. We start this section by checking these conditions for antiwebs.

LEMMA 5.1. *Let A be an (n, t) -antiweb and I_A the inequality $\sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$ for A . Then all edges of type not greater than $n \bmod t$ fulfill condition 2 of Proposition 3.5. All other edges (those of type greater than $n \bmod t$) violate condition 2.*

Proof. Consider a rim-edge of type $d \leq (n \bmod t)$, say $\{v_1, v_{d+1}\}$, and let $f = \lfloor n/t \rfloor$. Obviously, $ft + d \leq n$. We claim that the set $S = \{v_1\} \cup \{v_{d+1+t}, v_{d+1+2t}, \dots, v_{d+1+(f-1)t}\}$ is stable and fulfills condition 2 with respect to I . Regarding the stability it is easy to see that the second part of S is stable; it remains to verify that there is no edge between that part and v_1 . However, the latter is easy, as v_1 is adjacent only to its $t - 1$ successors (and they are not in S) and the $t - 1$ predecessors; for this notice that the last element $v_{d+1+(f-1)t}$ of S could only be in conflict with v_1 , but the neighbor of $v_{d+1+(f-1)t}$ being closest to v_1 is the vertex v_{d+ft} which is, as a consequence of $ft + d \leq n$, distinct from v_1 . Therefore S is stable. Regarding condition 2 notice that the neighbors of v_{d+1} which are not permitted to be in S are $(\{v_{d+1-(t-1)}, v_{d+1-(t-2)}, \dots, v_d\} \setminus \{v_1\}) \cup \{v_{d+2}, v_{d+3}, \dots, v_{d+t}\}$. The second part of the set is definitely not in S . For the vertices in the first part notice that if

$d + 1 - (t - 1) \leq 0$, then it corresponds to $d + 1 - (t - 1) + n \geq (f - 1)t + 2d + 2$ which is greater than $d + 1 + (f - 1)t$. However, on the other hand, if $d + 1 - (t - 1) > 0$ (that is, $d > t - 2$), then it follows together with $t > d$ that $d = t - 1$ and $d + 1 - (t - 1) = 1$, but we do not have to care about v_1 .

Finally, consider a rim-edge of type $d > (n \bmod t)$ and set $f = \lfloor n/t \rfloor$, and hence $ft + d > n$. Without loss of generality, assume that the rim-edge is the edge $\{v_1, v_{d+1}\}$. We want to construct a stable set $S \ni v_1$ which fulfills additionally condition 2. Hence some vertices cannot be in S , and the only choices besides v_1 for S belong to $\{v_{d+1+t}, v_{d+1+t+1}, \dots, v_{n+1-t}\}$. (The number of the last element is determined by the minimum distance of t from $v_1 = v_{n+1}$.) We need to choose $f - 1$ vertices; if there is a solution, then the remaining elements can be $\{v_{d+1+t}, v_{d+1+2t}, \dots, v_{d+1+(f-1)t}\}$. However, the vertex $v_{d+1+(f-1)t}$ does not belong to the set of candidates, as $d + 1 + (f - 1)t > n + 1 - t$. So this rim-edge of type $d > (n \bmod t)$ can never fulfill condition 2. \square

COROLLARY 5.2. *Let A be an (n, t) -antiweb with inequality $\sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$, k be a nonnegative integer, and $I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}}$ be the corresponding inequality for $G = A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$. Then all edges of A of type not greater than $n \bmod t$ fulfill condition 2 of Proposition 3.5 with respect to G . All other edges of A (those of type greater than $n \bmod t$) violate condition 2 for G .*

Proof. The proof is done by induction on k . Lemma 5.1 establishes the base-case of $k = 0$. So assume that the claim is proved for $(A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}, I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}})$. Lemma 4.6 establishes the claim then for $(A^{v_{0_1}, v_{0_2}, \dots, v_{0_{k+1}}}, I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_{k+1}}}})$. \square

We note that the corresponding inequality $I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}}$ for $A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$ is $\lfloor n/t \rfloor \sum_{i=1}^k x_{0_i} + \sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$.

LEMMA 5.3. *Let G be an (n, t) -antiweb with $n \not\equiv 0 \pmod t$ and let v be a vertex of G . For the facet $\sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$ the following two statements are equivalent:*

1. $n \equiv t - 1 \pmod t$.
2. v fulfills assumptions 1 and 2 of Theorem 3.7.

Proof. Let $k = \lfloor n/t \rfloor$. For the first implication (hence $n = kt + t - 1$) we can assume, without loss of generality (by symmetry), that $v = t$. Now we will construct the stable sets \tilde{S}_i for all neighbors $\{1, 2, \dots, t - 1, t + 1, t + 2, \dots, 2t - 1\}$ of i . Let $S' = \{3t - 1, 4t - 1, \dots, n\}$. Notice that $a^\top \chi^{S'} = k - 1$ and S contains no neighbor of $t, t + 1, \dots, 2t - 1$. Finally, for i with $t < i < 2t$ the sets $\tilde{S}_i = S' \cup \{i\}$ are stable and fulfill $a^\top \chi^{\tilde{S}_i} = k$. For the second half of necessary sets consider $S'' = \{2t, 3t, \dots, kt\}$ and use for i with $0 < i < t$ the sets $\tilde{S}_i = S'' \cup \{i\}$.

For the other direction we consider the vertex $v = t$. As the assumptions are fulfilled, there is a stable set \tilde{S} with $S \cap \{1, 2, \dots, 2t - 1\} = \{2t - 1\}$ and $a^\top \chi^{\tilde{S}} = k$. So \tilde{S} contains k elements. Choosing vertices of as small number as possible, \tilde{S} must be $\{2t - 1, 3t - 1, \dots, (k + 1)t - 1\}$. This requires that $n \geq (k + 1)t - 1$, and finally $n \equiv t - 1 \pmod t$. \square

Together, Lemmas 5.3 and 4.5 imply the following corollary.

COROLLARY 5.4. *Let A be an (n, t) -antiweb with $n \not\equiv 0 \pmod t$, let k be a nonnegative integer, and let v be a vertex of A . Consider $A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$. For the facet $I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}} : \lfloor n/t \rfloor \sum_{i=1}^k x_{0_i} + \sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$ of $STAB(A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}})$ the following two statements are equivalent:*

1. $n \equiv t - 1 \pmod t$.
2. v fulfills the assumptions 1 and 2 of Theorem 3.7.

THEOREM 5.5. *Let A be an (n, t) -antiweb with $n \not\equiv 0 \pmod t$ and let k be a nonnegative integer. Let $G = A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$, and the corresponding inequality is*

$I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}} : \lfloor n/t \rfloor \sum_{i=1}^k x_{0_i} + \sum_{i=1}^n x_i \leq \lfloor n/t \rfloor$. Then the following holds:

1. Every spoke can be facetly doubly subdivided.
2. Facetly star-subdivision at a vertex $v \in A$ is possible in G if and only if $n \equiv t - 1 \pmod t$.
3. An edge e of A can be facetly doubly subdivided in G if and only if its type is at most $n \pmod t$.

Proof. The first statement follows from successive application of Lemmas 4.7 and 4.6. The second statement is Corollary 5.4. The third statement is Corollary 5.2. \square

The next theorem gives a class of valid inequalities whose support graphs are the already introduced antiweb-wheels. For $s = 1, t = 2$, it reduces to $\mathcal{I}_{\mathcal{E}}$ given in [5]. The proof utilizes the three subdivision operations from the preceding two sections.

THEOREM 5.6 (validity of the (n, t) -antiweb- s -wheel inequality). *For $s \geq 0$ and an (n, t) -antiweb- s -wheel G the inequality*

$$(5.1) \quad \left\lfloor \frac{n}{t} \right\rfloor \sum_{i=1}^s x_{0_i} + \sum_{i \in \mathcal{O}} x_i + (2t - 3 + s) \sum_{i \in \mathcal{E}} x_i + \sum_{v \in \mathcal{S} \cup \mathcal{R}} x_v \leq \left\lfloor \frac{n}{t} \right\rfloor + (2t - 3 + s)|\mathcal{E}| + \frac{|\mathcal{S}| + |\mathcal{R}| - (2t - 2 + s)|\mathcal{E}|}{2}$$

is valid for $STAB(G)$, where \mathcal{S} denotes the set of internal vertices of the spoke-paths and \mathcal{R} denotes the set of internal vertices of the subdivided antiweb edges.

Proof. As a starting point, we utilize the validity of

$$(I_{A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}}) \quad \left\lfloor \frac{n}{t} \right\rfloor \sum_{i=1}^k x_{0_i} + \sum_{i=1}^n x_i \leq \left\lfloor \frac{n}{t} \right\rfloor$$

for $A^{v_{0_1}, v_{0_2}, \dots, v_{0_k}}$ from Theorem 5.5. Then we do star-subdivision at all vertices of \mathcal{E} ; here Theorem 3.7, part 1 guarantees that validity is maintained. The degree of every vertex in \mathcal{E} is $(2t - 2) + s$, where the first term accounts for the neighbors in the antiweb and the second term for the adjacent hub vertices. So if star-subdivision is applied at a vertex $v \in \mathcal{E}$, then $(2t - 2)$ new rim vertices and s spoke-vertices are added of weight 1. The weight of v is changed to $2t - 3 + s$ ($= \deg v - 1$), and the right-hand side is incremented by $2t - 3 + s$. This is accomplished in inequality (5.1) by the coefficient $2t - s + 3$ of $\sum_{i \in \mathcal{E}} x_i$, the term $\sum_{v \in \mathcal{S} \cup \mathcal{R}} x_v$, and for the right-hand side the term $(2t - 3 + s)|\mathcal{E}|$. Actually this operation subdivides also the spokes and cross-edges incident with v ; thereby the term $|\mathcal{S}| + |\mathcal{R}|$ on the right-hand side is increased by $(2t - 2 + s)|\mathcal{E}|$, so we need to subtract the same amount to balance this effect.

If in the antiweb-wheel that we want to reach there are paths of length 1 between members of \mathcal{E} , then the paths of length 3 between them can be doubly contracted by Proposition 3.6, part 1. Every contraction step changes the inequality in that the two terms x_u and x_w corresponding to vertices contracted away are dropped on the left-hand side of the inequality (as u, w are removed from \mathcal{S} or \mathcal{R}), and at the same time the right-hand side decreases by one.

If, finally, between some spoke-ends longer paths are necessary, they can be produced by applying edge-subdivision, which maintains validity according to Proposition 3.4. Again, the new inequality is of type (5.1). \square

As the question of validity is now settled for antiweb- s -wheels, we turn next to the question of facetness.

The proof of facetness is essentially the same as the preceding proof of validity, except that we require the facet version of each operation given in the preceding proof. Of course some of them have no corresponding counterparts.

THEOREM 5.7 (proper antiweb- s -wheel facets, $n \equiv t - 1 \pmod t$). *Given a proper antiweb- s -wheel G with $n \equiv t - 1 \pmod t$ the inequality (5.1) induces a facet of $STAB(G)$.*

We begin by giving an example to illustrate the proof of Theorem 5.7. We start with the inequality $2x_0 + \sum_{i=1}^8 x_i \leq 2$ which is facet-inducing for the graph in Figure 4.1. (Here $n = 8$ and $t = 3$). We want $\mathcal{E} = \{v_1, v_6, v_8\}$. By Theorem 5.5, part 2, each of v_1, v_2, \dots, v_8 can be facetly star subdivided, in particular v_1, v_6, v_8 . By Theorem 5.5, part 1, the spoke-edges

$$\{v_0, v_1\}, \{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\}, \{v_0, v_5\}, \{v_0, v_6\}, \{v_0, v_7\}, \{v_0, v_8\}$$

can be facetly doubly subdivided. By Theorem 5.5, part 3, the cross-edges

$$\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}, \{v_4, v_6\}, \\ \{v_5, v_6\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_6, v_8\}, \{v_7, v_8\}, \{v_7, v_1\}, \{v_8, v_1\}, \{v_8, v_2\}$$

can be facetly doubly subdivided. So apply Theorem 5.5, part 2 and facetly star subdivide v_1 . Hence the resulting inequality $2x_0 + 4x_1 + \sum_{i=2}^8 x_i + \sum_{i=1}^5 y_i \leq 6$ (we used y_i as variables for the vertices u_i , $i = 1, 2, 3, 4, 5$) is facet-inducing for the graph in Figure 4.3. Now, by Lemma 4.2, v_6 and v_8 can still be facetly star subdivided. Moreover, any original spoke-edges

$$\{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\}, \{v_0, v_5\}, \{v_0, v_6\}, \{v_0, v_7\}, \{v_0, v_8\}$$

and any original cross-edges

$$\{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}, \{v_4, v_6\}, \\ \{v_5, v_6\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_6, v_8\}, \{v_7, v_8\}, \{v_8, v_2\}$$

can still be facetly doubly subdivided by Lemma 4.4. We can now repeatedly apply Lemma 4.2 and facetly star subdivide at v_6 and v_8 successively. Hence the resulting inequality $2x_0 + 4x_1 + x_2 + x_3 + x_4 + x_5 + 4x_6 + x_7 + 4x_8 + \sum_{v \in \mathcal{S} \cup \mathcal{R}} x_v \leq 14$ is facet-inducing for the graph in Figure 4.4. Now, by repeatedly applying Lemma 4.4, any original spoke-edges $\{v_0, v_2\}, \{v_0, v_3\}, \{v_0, v_4\}, \{v_0, v_5\}, \{v_0, v_7\}$ and any original cross-edges $\{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}, \{v_5, v_7\}$ can still be facetly doubly subdivided. We now facetly doubly subdivide the remaining edges as necessary; that is, we want to replace each of $\{v_0, v_7\}, \{v_2, v_4\}, \{z, v_6\}$ (z is the vertex between v_0 and v_6) by a path of length 3 and to replace $\{v_3, v_5\}$ by a path of length 5. We first facetly doubly subdivide $\{v_0, v_7\}$. By Lemma 4.3, $\{v_2, v_4\}, \{v_3, v_5\}$, and $\{z, v_6\}$ can still be facetly doubly subdivided. We then facetly doubly subdivide $\{v_2, v_4\}$, so $\{v_3, v_5\}$ and $\{z, v_6\}$ can still be facetly doubly subdivided by Lemma 4.3. Now facetly doubly subdivide $\{v_3, v_5\}$ into a path $v_3 - h - k - v_5$. By Lemma 4.3, $\{h, k\}$ and $\{z, v_6\}$ can be facetly doubly subdivided, and we apply it to $\{h, k\}$. Now, since x_z has coefficient 1 and x_6 has coefficient 4 and x_z is of degree 2, we can facetly doubly subdivide $\{h, k\}$ by the remark after Proposition 3.5. Hence we have a facet-inducing inequality $2x_0 + 4x_1 + 4x_6 + 4x_8 + x_2 + x_3 + x_5 + x_7 + \sum_{v \in \mathcal{S} \cup \mathcal{R}} x_v \leq 19$ for Figure 4.2.

Proof of Theorem 5.7. The above example contains all the ingredients of the proof. Let G be the (n, t) -antiweb. We obtain the (n, t) -antiweb- s -wheel $G^{v_{0_1}, v_{0_2}, \dots, v_{0_s}}$ with

its facet-inducing inequality

$$\left\lfloor \frac{n}{t} \right\rfloor \sum_{i=1}^s x_{0_i} + \sum_{i \in G} x_i \leq \left\lfloor \frac{n}{t} \right\rfloor.$$

Consider an arbitrary wheel with prescribed set \mathcal{E} . By Theorem 5.5, part 2, every vertex G , in particular vertices in \mathcal{E} , can be facetly star subdivided. By Theorem 5.5, part 1, every spoke-edge can be facetly doubly subdivided. By Theorem 5.5, part 3, every cross-edge can be facetly doubly subdivided. So apply Theorem 5.5, part 2 and facetly star subdivide a vertex in \mathcal{E} (if such vertex exists). The resulting inequality is facet-inducing for the resulting graph. Now, by Lemma 4.2, each of the remaining vertices in \mathcal{E} can still be facetly star subdivided. Moreover, by Lemma 4.4 every original spoke-edge and every original cross-edge can still be facetly doubly subdivided as well as at least one of the two edges being created by a star-subdivision from one edge. We can now repeatedly apply Lemma 4.2 and facetly star subdivide each of the remaining vertices in \mathcal{E} successively. Hence the resulting inequality is facet-inducing for the resulting graph. Lemma 4.4 guarantees that on each spoke- and cross-path at least one edge can be facetly doubly subdivided. We now facetly doubly subdivide the remaining edges as necessary, which is possible by Lemmas 4.3 and 4.4. \square

THEOREM 5.8 (proper antiweb- s -wheel facets ($1 \leq (n \bmod t) \leq t - 2$)). *Given a proper antiweb- s -wheel G with $n \equiv r \pmod t$, $1 \leq a \leq t - 2$. Then inequality (5.1) induces a facet of the corresponding stable set polytope if and only if*

1. $\mathcal{E} = \emptyset$ and
2. all paths $P_{i,j}$ with $\{i, j\}$ of type $> r$ and $i, j \in \mathcal{O}$ have length 1.

Proof. The proof that 1 and 2 imply facetness is the same as that in Theorem 5.7, except that no star-subdivision is done, and only edges of type $\leq r$ are doubly subdivided (as necessary), while edges of type $> r$ are not doubly subdivided (as they do not fulfill the prerequisites of double edge subdivision).

For the other direction consider an arbitrary but proper (n, t) -antiweb- s -wheel G' with partition $\mathcal{E} \dot{\cup} \mathcal{O}$ that is facet-inducing for $\text{STAB}(G')$. First we want to apply Proposition 3.6, part 2 to shorten paths of length 3 to edges while maintaining facetness in the following way:

- Paths $P_{i,j}$ of type $> r$ and of length > 3 are shortened until their length is 2 or 3.
- Paths $P_{i,j}$ of type $< r$ with both ends in \mathcal{E} are shortened down to length 3.
- Paths $P_{i,j}$ of type $< r$ with at most one end in \mathcal{E} are shortened down to length 1 or 2.
- Spoke-paths are shortened down to length 1 or 2.

Denote with G the resulting graph. Now if G' violates 1 or 2, then G does, as we did not change \mathcal{E} at all, and even though we shortened the subdivided paths, we took care that paths violating 2 were not changed in any way. The resulting inequality induces a facet for $\text{STAB}(G)$.

Now consider a sequence of undoing star-subdivisions and shortening paths of length 3 to edges to reduce G to an unsubdivided (n, t) -antiweb-wheel H . Notice that by Lemma 3.11 the inequality after undoing a single star-subdivision is again facet-inducing. By Proposition 3.6, part 2, it follows similarly that the inequality after shortening a path is again facet-inducing.

Now consider the last intermediate graph H' in this sequence. By Theorem 5.5 we know that neither star-subdivision nor double edge subdivision are applicable to H . If the last operation is a star-subdivision, then by Lemma 3.10 it follows that going

from H to H' by star-subdivision destroys facetness; that is, the inequality for H' is not facet-inducing, contrary to assumption. If, on the other hand, the last operation is doubly subdividing an edge of type $> r$, then by Proposition 3.5 and the fact that we have a facet for H' we obtain that edge does not violate condition 2, giving again a contradiction. \square

So we obtained a complete characterization of the facet-inducing proper antiweb- s -wheels.

One might wonder how much more difficult a direct proof of the preceding theorem might be. We do believe that a direct proof would probably be shorter, though technically more involved. However, a direct proof of the “characterization” part would very likely be more difficult and substantially longer. Moreover, our algebraic approach to the interactions of the operations in section 3 and section 4 may be applicable to other problems.

6. Improper antiweb-wheels. The next natural question is the following: “How about the improper antiweb-wheels?” We do not know the complete answer to this, but we offer two preliminary results in this section.

The only difference between proper and improper antiweb-wheels is the requirement that for proper ones every path emanating from a vertex of \mathcal{E} must have length at least two. So every improper antiweb-wheel has a path between two members of \mathcal{E} of length 1 (instead of 3 for proper ones). We have never observed any improper antiweb-1-wheel facet. So we believe that they *are* indeed never facet-inducing.

As a stepping-stone for this puzzle we give the next lemma.

LEMMA 6.1 (the induced C_5). *Consider the stable set problem on a graph $G = (V, E)$ and a facet-inducing inequality $a^\top x \leq \alpha$. Let $G' = (V', E')$ be the support graph of $a^\top x \leq \alpha$ in G . If G' contains an induced 5-cycle $C = \{v_1, v_2, v_3, v_4, v_5\}$, where vertices v_2, v_4 have degree 2 in G , then for the vertex v_1 holds that its weight (with respect to a) is higher than the sum of the weights of its neighbors not in C .*

Proof. We prove the theorem by showing that if G contains such a 5-cycle and the weight condition is violated, then the face $a^\top x \leq \alpha$ is contained in the face induced by the 5-cycle inequality. So consider a stable set S with $a^\top \chi^S = \alpha$. Denote with b the characteristic vector of the odd 5-cycle C . We want to prove that $b^\top \chi^S = 2$ for all stable sets with $a^\top \chi^S = \alpha$. Suppose $a^\top \chi^S = \alpha$ and $b^\top \chi^S = 0$; then it is easy to see that the set $S' = S \cup \{v_2, v_4\}$ fulfills $a^\top \chi^{S'} > \alpha$, contradicting the validity of $a^\top x \leq \alpha$ for all stable sets.

Suppose next that $a^\top \chi^S = \alpha$ and $b^\top \chi^S = 1$. This requires $|C \cap S| = 1$. Without loss of generality, we can assume $C \cap S \subset \{v_1, v_2, v_3\}$. If $C \cap S \subset \{v_1, v_2\}$, then $S' = S \cup \{v_4\}$ violates $a^\top x \leq \alpha$. So it remains to study $C \cap S = \{v_3\}$. Either $S \cup \{v_1\}$ or $S \cup \{v_5\}$ is stable (and then violates $a^\top x \leq \alpha$) or both are not stable, because S contains neighbors of v_1 and v_5 outside of C . Now consider the set $S' = (S \setminus N(v_1)) \cup \{v_1\}$. Again, S' is stable. However, as the sum of the weights of the neighbors of v_1 outside of C is smaller than a_{v_1} we learn that $a^\top \chi^{S'} > a^\top \chi^S = \alpha$, contradicting the validity of $a^\top x \leq \alpha$. \square

This lemma helps to weed out many of the antiweb-wheels that are not facet-inducing, as demonstrated by the following corollary.

COROLLARY 6.2. *Let G be an improper antiweb- s -wheel ($s \geq 1$) with partition \mathcal{E}, \mathcal{O} and corresponding antiweb-1-wheel inequality I_G . Let $H = G[\mathcal{E}]$. If H contains a vertex of degree 1, then I_G is not facet-inducing.*

Proof. Consider a vertex $u \in \mathcal{E}$ of degree 1 and its unique neighbor $w \in \mathcal{E}$ and assume that the inequality I_G nevertheless induces a facet. Without loss of generality,

we can assume that the spoke-paths from u, w to the hub have length two (otherwise we could shorten them with Proposition 3.6, part 2). Consider the C_5 induced by u, w , the edge between them and their two spoke paths. Now notice that u has weight $2t - 3 + s$ and it has only $(2t - 2 + s) - 2$ neighbors outside of this C_5 ; all of these $(2t - 2 + s) - 2$ neighbors have weight 1. So we might conclude that by the Lemma 6.1 the valid inequality does not define a facet, contrary to assumption. \square

7. Concluding remarks. Using well-known transformations, one gets the valid inequalities corresponding to the new inequalities of this study for the cut polytope, as performed for the wheel inequalities, for example, by Cheng [4]; from the cut polytope they carry over to the boolean quadric polytope as demonstrated by De Simone [17]. Similarly, they could be utilized for the multiwaycut problem [2].

In an accompanying paper [8] we give efficient separation algorithms for generalizations of t -antiweb- s -wheel inequalities for fixed t and s . These generalizations (nonsimple antiweb-wheels) permit the identification of nonadjacent vertices outside the hub. They were introduced only to facilitate the design of our algorithms.

Acknowledgment. We would like to thank the anonymous referees for their valuable comments.

REFERENCES

- [1] F. BARAHONA AND A. R. MAHJOUR, *Compositions of graphs and polyhedra II: Stable sets*, SIAM J. Discrete Math., 7 (1994), pp. 359–371.
- [2] D. BERTISMAS, C. TEO, AND R. V. VOHRA, *Nonlinear formulations and improved randomized approximation algorithms for multicut problems*, in Proceedings of the Fourth International Integer Programming and Combinatorial Optimization Conference, Copenhagen, Denmark, Lecture Notes in Comput. Sci. 920, E. Balas and J. Clausen, eds., Springer-Verlag, Berlin, Heidelberg, 1995, pp. 29–39.
- [3] E. CHENG, *Wheel Inequalities for Stable Set Polytopes*, Ph.D thesis, University of Waterloo, Waterloo, ON, Canada, 1995.
- [4] E. CHENG, *Separating subdivision of bicycle wheel inequalities over cut polytopes*, Oper. Res. Lett., 23 (1998), pp. 13–19.
- [5] E. CHENG AND W. H. CUNNINGHAM, *Wheel inequalities for stable set polytopes*, Math. Programming, 77 (1997), pp. 389–421.
- [6] E. CHENG AND S. DE VRIES, *Antiweb-Wheel Inequalities for Stable Set Polytopes: Separation and Facetness*, Technical report 99-3, Department of Mathematics and Statistics, Oakland University, Rochester, MI, 1999.
- [7] E. CHENG AND S. DE VRIES, *Antiweb inequalities: Strength and intractability*, Congr. Numer., 152 (2001), pp. 5–19.
- [8] E. CHENG AND S. DE VRIES, *Antiweb-wheel inequalities and their separation problems over the stable set polytopes*, Math. Program., 92 (2002), pp. 153–175.
- [9] V. CHVÁTAL, *On certain polytopes associated with graphs*, J. Combinatorial Theory Ser. B, 18 (1975), pp. 138–154.
- [10] G. DAHL, *Stable set polytopes for a class of circulant graphs*, SIAM J. Optim., 9 (1999), pp. 493–503.
- [11] S. DE VRIES, *Discrete Tomography, Packing and Covering, and Stable Set Problems: Polytopes and Algorithms*, Ph.D. thesis, Technische Universität München, München, Germany, 1999.
- [12] R. EULER, M. JÜNGER, AND G. REINELT, *Generalizations of cliques, odd cycles and anticliques and their relation to independence system polyhedra*, Math. Oper. Res., 12 (1987), pp. 451–462.
- [13] M. C. GOLUBIC AND P. L. HAMMER, *Stability in circular arc graphs*, J. Algorithms, 9 (1988), pp. 314–320.
- [14] M. LAURENT, *A generalization of antiwebs to independence systems and their canonical facets*, Math. Programming, 45 (1989), pp. 97–108.
- [15] R. MÜLLER AND A. S. SCHULZ, *Transitive packing*, in Proceedings of the Fifth International Integer Programming and Combinatorial Optimization Conference, Vancouver, Canada, Lec-

- ture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer-Verlag, Berlin, Heidelberg, 1996, pp. 430–444.
- [16] A. S. SCHULZ, *Polytopes and Scheduling*, Ph.D. thesis, TU Berlin, Berlin, Germany, 1996.
 - [17] C. DE SIMONE, *The cut polytope and the boolean quadric polytope*, Discrete Math., 79 (1989/90), pp. 71–75.
 - [18] L. E. TROTTER, *A class of facet producing graphs for vertex packing polyhedra*, Discrete Math., 12 (1975), pp. 373–388.
 - [19] L. A. WOLSEY, *Further facet generating procedures for vertex packing polytopes*, Math. Programming, 11 (1976), pp. 158–163.

CHARACTERIZATION OF EFFICIENTLY PARALLEL SOLVABLE PROBLEMS ON DISTANCE-HEREDITARY GRAPHS*

SUN-YUAN HSIEH[†], CHIN-WEN HO[‡], TSAN-SHENG HSU[§], MING-TAT KO[§], AND
GEN-HUEY CHEN[¶]

Abstract. In this paper, we sketch common properties of a class of so-called subgraph optimization problems that can be systematically solved on distance-hereditary graphs. Based on the found properties, we then develop a general problem-solving paradigm that solves these problems efficiently in parallel. As a by-product, we also obtain new linear-time algorithms by a sequential simulation of our parallel algorithms. Let $T_d(|V|, |E|)$ and $P_d(|V|, |E|)$ denote the time and processor complexities, respectively, required to construct a decomposition tree of a distance-hereditary graph $G = (V, E)$ on a PRAM model M_d . Based on the proposed paradigm, we show that the maximum independent set problem, the maximum clique problem, the vertex connectivity problem, the domination problem, and the independent domination problem can be sequentially solved in $O(|V| + |E|)$ time, and solved in parallel in $O(T_d(|V|, |E|) + \log |V|)$ time using $O(P_d(|V|, |E|) + |V|/\log |V|)$ processors on M_d . By constructing a decomposition tree under a CREW PRAM, we also show that $T_d(|V|, |E|) = O(\log^2 |V|)$ and $P_d(|V|, |E|) = O(|V| + |E|)$.

Key words. algorithms, data structures, distance-hereditary graphs, parallel random access machine, subgraph optimization problems

AMS subject classifications. 68P05, 68Q25, 68R10, 68W10, 68W40

PII. S0895480101389880

1. Introduction. A graph is *distance-hereditary* [2, 18] if the distance stays the same between any of two vertices in every connected induced subgraph containing both (where the *distance* between two vertices is the length of a shortest path connecting them). Distance-hereditary graphs form a subclass of perfect graphs [11, 15, 18] that are graphs G in which the maximum clique size equals the chromatic number for every induced subgraph of G [3, 13]. Two well-known classes of graphs, trees and cographs, both belong to distance-hereditary graphs. There were sequential or parallel algorithms to solve quite a few interesting graph-theoretical problems on this special class of graphs. The interested readers may consult [2, 5, 6, 11, 12, 15, 16, 18, 19, 20, 25, 27, 28, 29] for details.

*Received by the editors May 25, 2001; accepted for publication (in revised form) June 20, 2002; published electronically September 10, 2002. Two various parts of this paper appeared in *Proceedings of the 9th International Symposium on Algorithms and Computation (ISAAC'98)*, *Lecture Notes in Comput. Sci.* 1533, 1998, pp. 257–266, and in *Proceedings of the 4th International ACPC Conference Including Special Tracks on Parallel Numerics (ParNum'99) and Parallel Computing in Image Processing, Video Processing, and Multimedia (ACPC'99)*, *Lecture Notes in Comput. Sci.* 1557, 1999, pp. 417–426.

<http://www.siam.org/journals/sidma/15-4/38988.html>

[†]Department of Computer Science and Information Engineering, National Cheng Kung University, 1 University Rd., Tainan 701, Taiwan (hsiehsy@mail.ncku.edu.tw). The work of this author was supported in part by the National Science Council under grant NSC 89-2218-E-260-015 and by the Institute of Information Science, Academia Sinica, Taiwan. Part of this work was done while this author was visiting Academia Sinica, Taiwan.

[‡]Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan (hocw@csie.ncu.edu.tw). The work of this author was supported in part by the National Science Council under grant NSC 88-2213-E-008-001.

[§]Institute of Information Science, Academia Sinica, Taipei, Taiwan (tshsu@iis.sinica.edu.tw, mtko@iis.sinica.edu.tw).

[¶]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (ghchen@csie.ntu.edu.tw).

Several characterizations of distance-hereditary graphs were also explored for algorithmic applications. In [2], Bandelt and Mulder showed that the house, holes, domino, and gem are neither induced subgraphs nor isometric subgraphs of a distance-hereditary graph. In [15], Hammer and Maffray utilized the hanging structure to show that a graph is distance-hereditary if and only if it has a one-vertex-extension ordering. Using this ordering, they proposed a linear $O(|V| + |E|)$ -sequential-time recognition algorithm, where V and E are the vertex and edge sets of the given graph. The vertex-coloring problem and the maximum weighted stable set problem were also solved in linear time in [15]. In [6], Chang, Hsieh, and Chen generalized the concept of the one-vertex-extension ordering to define the one-vertex-extension tree. They further obtained a new recursive definition of distance-hereditary graphs and showed that this new characterization can be utilized to solve the weighted vertex cover problem, the weighted independent domination problem, the minimum fill-in problem, and the tree-width problem. The former (respectively, latter) two problems need $O(|V| + |E|)$ (respectively, $O(|V|^2)$) sequential time. Quite recently, Golumbic and Rotics [14] showed that distance-hereditary graphs are those graphs of clique-width at most three for which a corresponding 3-expression can be built in linear sequential time. Moreover, Courcelle, Makowsky, and Rotics [9] showed an elegant result that given a k -expression of a graph G with the bounded clique-width k , all graph problems expressible in monadic second order logic with quantification over vertex sets only can be solved in linear time on G . Therefore, a wide class of graph problems are linear-time solvable on distance-hereditary graphs.

Most known polynomial time algorithms on distance-hereditary graphs utilize techniques discovered from the properties of the problems and graphs, which we feel are inherently sequential. In this paper, we propose a new approach based on the one-vertex-extension tree proposed in [6] to come out a general problem-solving paradigm, and thus a good structure for representing distance-hereditary graphs, for designing parallel algorithms for a class of problems on distance-hereditary graphs. Note that we also obtain linear-time algorithms that are different from the previous studies of other researchers for all these problems by sequentially simulating our parallel algorithms. Given a graph problem, we say it belongs to the class of *subgraph optimization* problem if the object of this problem is to find a subgraph of the input graph to satisfy the given properties which include an optimization constraint. For example, the problem of finding a maximum independent set is a subgraph optimization problem. By discovering recursive properties of distance-hereditary graphs, we define a general problem-solving paradigm for subgraph optimization problems. The paradigm consists of the two main phases. The first phase is to construct a binary tree structure, called a *decomposition tree*, for representing a distance-hereditary graph. The second phase is to reduce the given subgraph optimization problem to another problem which can be solved on a decomposition tree. Problems that fit in our paradigm include the following: (a) the maximum clique problem, (b) the maximum independent set problem, (c) the vertex connectivity problem, (d) the domination problem, and (e) the independent domination problem. All the above problems but problem (c) were shown to be linear-time solvable [6, 9, 14, 15].

Let $T_d(|V|, |E|)$ and $P_d(|V|, |E|)$ denote the time complexity and processor complexity required to construct a decomposition tree of a distance-hereditary graph $G = (V, E)$ on a PRAM model M_d . We show that problems (a)–(e) can be sequentially solved in $O(|V| + |E|)$ time, and solved in parallel in $O(T_d(|V|, |E|) + \log |V|)$ time using $O(P_d(|V|, |E|) + |V|/\log |V|)$ processors on M_d . If a decomposition tree

is given to be the input instance, problems (a)–(e) can be solved in $O(\log |V|)$ time using $O(|V|/\log |V|)$ processors on an EREW PRAM. To our knowledge, the sequential complexity of problem (c) and the parallel complexities of problems (b)–(e) remains unknown in the literatures. Note that previous known parallel complexities for problem (a) on distance-hereditary graphs were $O(\log^2 |V|)$ time using $O(|V|+|E|)$ processors on a CREW PRAM [19]. For the rest, we match the current best algorithms [6, 15, 19]. By constructing a decomposition tree in parallel, we also show that $T_d(|V|, |E|) = O(\log^2 |V|)$, $P_d(|V|, |E|) = O(|V| + |E|)$ under a CREW PRAM.

The computation model used here is the deterministic parallel random access machine (PRAM) which permits concurrent read and exclusive write (CREW), or exclusive read and write (EREW) in its shared memory [22]. The rest of this paper is organized as follows. In section 2, we review some properties of distance-hereditary graphs and give basic definitions. In section 3, we define a general problem-solving paradigm and develop its sequential and parallel implementation. In section 4, we show that problems (a)–(e) are examples that fit into our paradigm. In section 5, we present a parallel algorithm to construct a decomposition tree for a distance-hereditary graph. Finally, some concluding remarks are given in section 6.

2. Preliminaries. This paper considers finite, simple, and undirected graphs $G = (V, E)$, where V and E are the vertex and edge sets of G , respectively. Let $n = |V|$ and $m = |E|$. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *union of G_1 and G_2* , denoted by $G_1 \cup G_2$, is the graph $(V_1 \cup V_2, E_1 \cup E_2)$. Let $G[X]$ denote the subgraph of G induced by $X \subseteq V$. For graph-theoretic terminologies and notations not mentioned here, see [13]. For a vertex $v \in V$ of a graph $G = (V, E)$, the *neighborhood* of v is $N_G(v) = \{u \in V \mid (u, v) \in E\}$ and the *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. We use $N(v)$ for $N_G(v)$, and $N[v]$ for $N_G[v]$, if there is no ambiguity.

For a graph $G = (V, E)$, the *degree* of a vertex $v \in V$ is $\deg(v) = |N(v)|$. We say that vertex u is a *pendant vertex* attached to vertex v if $\deg(u) = 1$ and v is the vertex adjacent to u . Two vertices u and v are called *true* (respectively, *false*) *twins* if $N[u] = N[v]$ (respectively, $N(u) = N(v)$).

Given a graph $G = (V, E)$, an ordering $\delta = (v_1, v_2, \dots, v_n)$ of V is said to be a *one-vertex-extension ordering* of G if v_i is a pendant vertex attached to some vertex in $G[V_i]$ or is a twin of some vertex in $G[V_i]$ for $1 \leq i \leq n$, where $V_i = \{v_1, v_2, \dots, v_i\}$.

LEMMA 2.1 (see [2, 15]). *A graph is distance-hereditary if and only if it has a one-vertex-extension ordering.*

Let $G = (V, E)$ be a distance-hereditary graph with a one-vertex-extension-ordering $\delta = (v_1, v_2, \dots, v_n)$. In [6], Chang, Hsieh, and Chen constructed a *one-vertex-extension tree*, denoted by \mathcal{E}_G , with respect to δ as follows. Tree \mathcal{E}_G is a rooted ordered tree rooted at v_1 with the node set V . For $j = 2, 3, \dots, n$, we let v_j be the rightmost child of v_i , $i < j$, in the current tree if either v_j is a pendant vertex attached to v_i or v_j and v_i are twins in $G[V_j]$. We use (v_j, v_i) to denote an edge of \mathcal{E}_G . Moreover, (v_j, v_i) is labelled with P if v_j is a pendant vertex attached to v_i in $G[V_j]$, and it is labelled with T (respectively, F) if v_i and v_j are true twins (respectively, false twins) in $G[V_j]$.

LEMMA 2.2 (see [6]). *A one-vertex-extension tree of a distance-hereditary graph can be constructed in $O(n + m)$ time.*

Figure 2.1(a) shows a distance-hereditary graph whose vertex set is associated with a one-vertex-extension ordering. Figure 2.1(b) shows a one-vertex-extension tree with respect to the ordering.

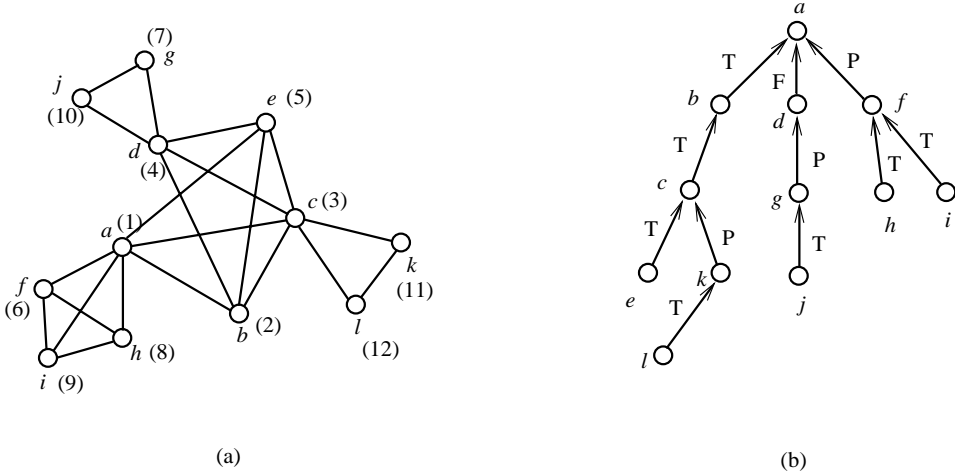


FIG. 2.1. A distance-hereditary graph and its one-vertex-extension tree. The numbers (1)–(12) associated with the vertices of the graph shown in (a) correspond to a one-vertex-extension ordering.

The twin set of $v \in V(\mathcal{E}_G)$, denoted by $S(v)$, consists of v and the descendants of v such that v can be reached through only T or F edges. The twin set of \mathcal{E}_G (or twin set of G) is the twin set of the root of \mathcal{E}_G . In Figure 2.1(b), the twin set of G is $\{a, b, c, d, e\}$.

Suppose nodes $v_{i_1} < v_{i_2} < \dots < v_{i_{j-1}} < v_{i_j} < v_{i_{j+1}} < \dots < v_{i_k}$ are children of v_i in \mathcal{E}_G . For an edge (v_{i_j}, v_i) in \mathcal{E}_G , let $S_r(v_{i_j}, v_i) = S(v_i) \setminus (\cup_{l=1}^j S(v_{i_l}))$. Let $\mathcal{E}_G(v_{i_j}, v_i)$ denote the subtree of \mathcal{E}_G induced by $v_i, v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}$ and all descendants of $v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}$. Recall that $\mathcal{E}_G(v)$ is used to denote the subtree rooted at v in \mathcal{E}_G .

We say that two disjoint vertex subsets X and Y form a join in a graph $G = (V, E)$ if every vertex of X is connected to every vertex of Y .

LEMMA 2.3 (see [6]). Suppose that v_j is a child of v_i in \mathcal{E}_G . Then the following two statements hold.

1. If (v_j, v_i) is labelled with P or T, then $S(v_j)$ and $S_r(v_j, v_i)$ form a join in G . Moreover, for every vertex $v \in V(\mathcal{E}_G(v_j)) \setminus S(v_j)$, $N[v] \subseteq V(\mathcal{E}_G(v_j))$.
2. If (v_j, v_i) is labelled with F, then every vertex of $V(\mathcal{E}_G(v_j))$ is not adjacent to any vertex of $V(\mathcal{E}_G(v_j, v_i)) \setminus V(\mathcal{E}_G(v_j))$ in G .

Given a distance-hereditary graph $G = (V, E)$, there exists a one-vertex-extension ordering (v_1, v_2, \dots, v_n) . This ordering corresponds to a one-vertex-extension tree \mathcal{E}_G . Note that the twin set of G is $S(v_1)$. The vertex set V can be partitioned into four disjoint sets: $V(\mathcal{E}_G(v_2)) \setminus S(v_2)$, $S(v_2)$, $(V \setminus V(\mathcal{E}_G(v_2))) \setminus S_r(v_2, v_1)$, and $S_r(v_2, v_1)$. By Lemma 2.3, G can be regarded as to be formed from $G_1 = G[V \setminus V(\mathcal{E}_G(v_2))]$ and $G_2 = G[V(\mathcal{E}_G(v_2))]$ by the three operations according to the type (v_2, v_1) in \mathcal{E}_G as follows. If (v_2, v_1) is labelled T or P, then G is formed from G_1 and G_2 by connecting every vertex of $S_r(v_2, v_1)$ to all vertices of $S(v_2)$. If (v_2, v_1) is labelled F, then G is the union of G_1 and G_2 . If (v_2, v_1) is labelled P, then the twin set of G is the twin set of G_1 . If $[v_1, v_2]$ is labelled T or F, then the twin set of G is the union of the twin set of G_1 and G_2 . Based upon the above observations, we provide a characterization for distance-hereditary graphs below.

A graph consisting of a single vertex v is clearly a distance-hereditary graph. It is said to be a *primitive distance-hereditary graph with the twin set* $\{v\}$ [6]. A graph G with $|V(G)| \geq 2$ is distance-hereditary if and only if it can be obtained by three operations described in the following lemma. Let G_1 and G_2 be distance-hereditary graphs with the twin sets S_1 and S_2 , respectively.

LEMMA 2.4 (see [6]). 1. *The graph obtained from G_1 and G_2 by connecting every vertex of S_1 to all vertices of S_2 is a distance-hereditary graph with the twin set $S_1 \cup S_2$.*

2. *The graph obtained from G_1 and G_2 by connecting every vertex of S_1 to all vertices of S_2 is a distance-hereditary graph with the twin set S_1 .*

3. *The union of G_1 and G_2 is a distance-hereditary graph with the twin set $S_1 \cup S_2$.*

Note that the difference between operations 1 and 2 of Lemma 2.4 is the twin set construction.

A distance-hereditary graph G is said to be formed from G_1 with the twin set S_1 and G_2 with the twin set S_2 by the *true twin* (respectively, *attachment*) operation if G is obtained through operation 1 (respectively, 2) of Lemma 2.4, and by the *false twin operation* if G is obtained through operation 3 of Lemma 2.4.

A distance-hereditary graph can be represented by a binary tree form, called a *decomposition tree*, which is defined as follows.

DEFINITION 2.5 (see [6]). 1. *The tree consisting of a single vertex v is a decomposition tree of the primitive distance-hereditary graph $G = (\{v\}, \emptyset)$.*

2. *Let \mathcal{D}_1 and \mathcal{D}_2 be the decomposition trees of distance-hereditary graphs G_1 and G_2 , respectively.*

(a) *If G is formed from G_1 and G_2 by the true twin operation, then a tree \mathcal{D} with the root r represented by \otimes and with the roots of \mathcal{D}_1 and \mathcal{D}_2 being the two children of r is a decomposition tree of G .*

(b) *If G is formed from G_1 and G_2 by the attachment operation, then a tree \mathcal{D} with the root r represented by \oplus and with the roots of \mathcal{D}_1 and \mathcal{D}_2 being the left child and the right child of r , respectively, is a decomposition tree of G .*

(c) *If G is formed from G_1 and G_2 by the false twin operation, then a tree \mathcal{D} with the root r represented by \odot and with the roots of \mathcal{D}_1 and \mathcal{D}_2 being the two children of r is a decomposition tree of G .*

Figure 2.2 shows a distance-hereditary graph and its decomposition tree. Note that the twin set of the given graph is $\{a, b, c, d\}$.

LEMMA 2.6. *A decomposition tree of a distance-hereditary graph can be constructed in $O(n + m)$ sequential time.*

Proof. It follows from the fact that a one-vertex-extension tree can be generated in $O(n + m)$ time [6]. \square

3. A general problem-solving paradigm.

3.1. The subgraphs generating problem. Suppose that $G = (V, E)$ is a graph and let \mathcal{U} be the set consisting of all subsets of V . Given a set $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_l\}$, where $Q_i \in \mathcal{U}$, we define MIN_v to be an operator on \mathcal{Q} that returns a set Q_j , for some $1 \leq j \leq l$, such that $|Q_j|$ is the smallest. The operator MAX_v is defined similarly. Given $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_l\}$ and $\mathcal{R} = \{R_1, R_2, \dots, R_t\}$, where $\mathcal{Q}, \mathcal{R} \subset \mathcal{U}$, \mathcal{Q} and \mathcal{R} are *disjoint* if $Q_i \cap R_j = \emptyset$, $1 \leq i \leq l$, and $1 \leq j \leq t$. For two lists $L_1 = \langle l_1, l_2, \dots, l_k \rangle$ and $L_1' = \langle l_1', l_2', \dots, l_j' \rangle$, we define the *concatenation of L_1 and L_1'* , denoted by $L_1 \bullet L_1'$, to be the list $\langle l_1, l_2, \dots, l_k, l_1', l_2', \dots, l_j' \rangle$.

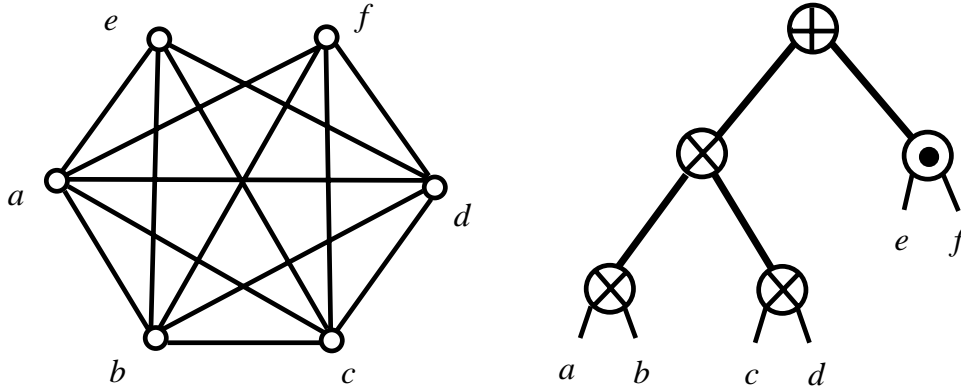


FIG. 2.2. A distance-hereditary graph with its decomposition tree.

Consider a rooted tree T . Let $root(T)$ be the root of T . For a node x in T , any node y on the unique path from x to $root(T)$ is called an *ancestor* of x . If y is an ancestor of x , then x is a *descendant* of y . Further, x is a *proper descendant* of y when $x \neq y$. Note that every node is both an ancestor and a descendant of itself. Two nodes in T are *irrelative* if one is not an ancestor of the other. The *least common ancestor* of two nodes x and y in T is the node that is an ancestor to both x and y , and is farthest from $root(T)$.

DEFINITION 3.1. Let $G = (V, E)$ be a graph and let T be a binary tree. Also let \mathcal{U} be the set consisting of all subsets of V . Given two nonnegative integers k and r , and an operator $\Theta \in \{\text{MIN}_v, \text{MAX}_v\}$, T is an (r, k, Θ) -subgraph generating tree of G if the following conditions hold. Let v be a node of T and let N_i be the set of integers from 1 to i .

1. Node v is associated with a list of r subgraphs $A_v = \langle A_{v,1}, A_{v,2}, \dots, A_{v,r} \rangle$ selected from \mathcal{U} such that $|A_{v,i}| = O(1)$ and A_v and A_w are disjoint if v and w are irrelative. These subgraphs are called the auxiliary subgraphs¹ of v .
2. If v is an internal node, then it is associated with k integers $a_{v,1}, a_{v,2}, \dots, a_{v,k}$ from N_{r+k} , and the following $2k$ linear unary functions $f_{v,i} : N_{a_{v,i}} \mapsto N_{r+k}$ and $g_{v,i} : N_{a_{v,i}} \mapsto N_{r+k}$, $1 \leq i \leq k$.
3. Node v is also associated with a list of k subgraphs $R_v = \langle R_{v,1}, R_{v,2}, \dots, R_{v,k} \rangle$, called the target subgraphs of v , which are defined as follows.
 - (a) If v is a leaf, then R_v is a list of subgraphs selected from \mathcal{U} . Moreover, R_x and R_y are disjoint for two arbitrary distinct leaves x and y .
 - (b) If v is an internal node with the children u and w , then

$$(3.1) \quad R_{v,i} = \Theta \{ Z_{u,f_{u,i}(1)} \cup Z_{w,g_{w,i}(1)}, Z_{u,f_{u,i}(2)} \cup Z_{w,g_{w,i}(2)}, \dots, Z_{u,f_{u,i}(a_{v,i})} \cup Z_{w,g_{w,i}(a_{v,i})} \},$$

where $1 \leq i \leq k$, $Z_u = R_u \bullet A_u = \langle Z_{u,1}, Z_{u,2}, \dots, Z_{u,k+r} \rangle$, $Z_w = R_w \bullet A_w = \langle Z_{w,1}, Z_{w,2}, \dots, Z_{w,k+r} \rangle$. Note that $Z_{u,f_{u,i}(j)} \cap Z_{w,g_{w,i}(j)} = \emptyset$ for $1 \leq j \leq a_{v,i}$.

For a node v in an (r, k, Θ) -subgraph generating tree T , let $T(v)$ be a subtree of T rooted at v and let G_v be the subgraph of G induced by the leaves of $T(v)$. Also let

¹For ease of implementation, we allow a subgraph of G to be represented by its vertex set if it has no edge.

\mathcal{U}_v be the set consisting of all subsets of $(\cup_{1 \leq i \leq k} R_{x,i}) \cup (\cup_{1 \leq i \leq r} A_{y,i})$, where x is a leaf of $T(v)$ and $y \in V(T(v))$. Note that \mathcal{U}_v and \mathcal{U}_w are disjoint if v and w are irrelative.

Let G be a distance-hereditary graph. As we will show in sections 4.1–4.5, solving some subgraph optimization problem \mathcal{P} on G can be transformed easily into solving that on a corresponding (r, k, Θ) -subgraph generating tree T of G . According to the essential property of \mathcal{P} , each node $v \in V(T)$ can be associated with $r(\geq 0)$ subgraphs A_v of G_v in advance such that the following condition holds. For an internal node v with two children u and w , a solution of \mathcal{P} on G_v can be obtained by selecting a subgraph with the maximum or minimum cardinality (depending on \mathcal{P}) from the $2(k+r)$ subgraphs in some combinations of R_u, R_w, A_u , and A_w shown as (3.1). Note that R_v is generated in a bottom-up fashion, and the selection can be implemented according Θ together with $f_{u,i}, f_{w,i}, g_{u,i}$ and $g_{w,i}$, $1 \leq i \leq k$.

DEFINITION 3.2. *Let T be an (r, k, Θ) -subgraph generating tree. The (r, k, Θ) -subgraph generating problem is the problem of finding the k target subgraphs of the root of T .*

LEMMA 3.3. *The (r, k, Θ) -subgraph generating problem can be solved in $O((rk + k^2)n)$ time, where n is the number of vertices of the given tree.*

Proof. Clearly, the problem can be solved by a bottom-up evaluation of the given tree. We now show the complexity. Note that there are $l \leq r + k$ subgraphs in (3.1) to generate $R_{v,i}$, $1 \leq i \leq k$, using Θ . Without loss of generality, assume that v is an internal node. According to (3.1), each term is obtained by the union of two disjoint sets selected using the functions $f_{u,i}$ and $g_{w,i}$, where u and w are the two children of v . Since both $f_{u,i}$ and $g_{w,i}$ are linear unary functions which can be evaluated in $O(1)$ time, the desired l subgraphs can be obtained in $O(r + k)$ time. Next, we explain how to implement Θ to select a target subgraph. We can record the cardinality of each of l subgraphs such that generating $R_{v,i}$ is equivalent to finding the maximum (or minimum) among l values. This can be implemented to run in $O(r + k)$ time. Therefore, generating $R_{v,i}$'s for all $1 \leq i \leq k$ takes $O(rk + k^2)$ time. Since there are totally n vertices in the tree, the problem can be solved with the desired complexity. \square

3.2. Parallel complexities of the (r, k, Θ) -subgraph generating problem.

In this section, we apply the binary tree contraction technique described in [1] to parallelize the (r, k, Θ) -subgraph generating problem. This technique recursively applies two operations, *prune* and *bypass*, to a given binary tree. *Prune*(u) is an operation which removes a leaf node u from the current tree, and *bypass*(v) is an operation (following a prune operation) that removes a node v with exactly one child w and then lets the parent of v become the new parent of w . We define a *contraction phase* to be the consecutive execution of a prune and then bypass operations. Figure 3.1 shows two procedures, *prune*(u) and *bypass*(v).

Let T be an n -leave binary tree. Given an Euler tour starting from $root(T)$ of T , the algorithm initially numbers the leaves from 1 to n according to the order of their appearances in the tour. Then the algorithm repeats the following steps. In each step, *prune* and *bypass* work only on the leaves with odd indices and their parents. Hence, these two operations can be performed independently and delete $\lfloor \frac{l}{2} \rfloor$ leaves together with their parents on the binary tree in each step, where l is the number of the current leaves. Therefore, the tree will be reduced to a three-node tree after repeating the steps in $\lceil \log n \rceil$ times.

LEMMA 3.4 (see [1]). *If the prune operation and bypass operation can be performed by one processor in constant time, the binary tree contraction algorithm can*

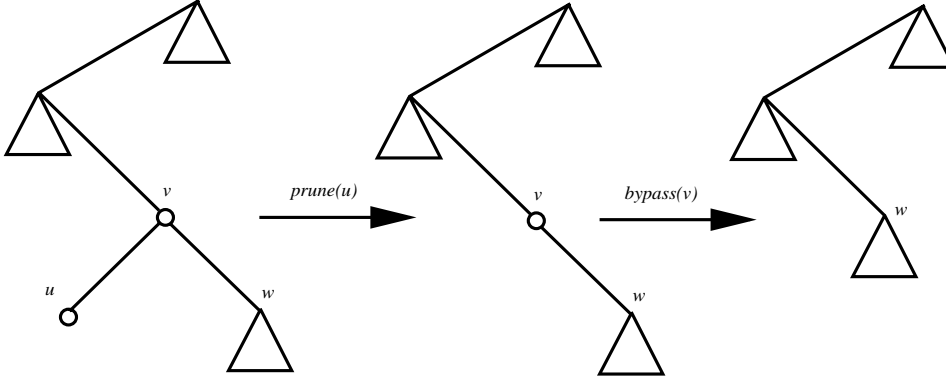


FIG. 3.1. Illustrating two procedures, $prune(u)$ and $bypass(v)$.

be implemented in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM, where n is the number of nodes in an input binary tree.

DEFINITION 3.5. Let u and v be two nodes of an (r, k, Θ) -subgraph generating tree T such that u is a descendant of v . A k -ary function $h : \mathcal{U}_u^k \mapsto \mathcal{U}_v$ possesses the closed form if $h(X_1, \dots, X_k) = \Theta\{X_{b_1} \cup C_1, X_{b_2} \cup C_2, \dots, X_{b_a} \cup C_a, Q\}$, where $b_i \neq b_j$ for two distinct $1 \leq i, j \leq a$, and $C_i, Q \in (\mathcal{U}_v \setminus \mathcal{U}_u)$.

LEMMA 3.6. Let $\Theta \in \{\text{MIN}_v, \text{MAX}_v\}$, and let $h_0 : \mathcal{U}_u^k \mapsto \mathcal{U}_v$ be a function with the closed form, where u is a descendant of v . Let w be a descendant of u and let $h_i : \mathcal{U}_w^k \mapsto \mathcal{U}_u$, $1 \leq i \leq k$, be k functions possessing the closed form. Then the function obtained from the composition $h_0 \circ (h_1, h_2, \dots, h_k) : \mathcal{U}_w^k \mapsto \mathcal{U}_v$ also possesses the closed form.

Proof. Let $h_i(X_1, \dots, X_k) = \Theta\{X_{b_1^i} \cup C_1^i, X_{b_2^i} \cup C_2^i, \dots, X_{b_{a_i}^i} \cup C_{a_i}^i, Q_i\}$ for all $0 \leq i \leq k$. Note that $C_j^i, Q_i \in (\mathcal{U}_u \setminus \mathcal{U}_w)$, $1 \leq j \leq a_i$. Define function $B(i, j) = b_j^i$, where $0 \leq i \leq k$ and $1 \leq j \leq a_i$. We show in the following that $h_0 \circ (h_1, \dots, h_k)$ is a function with the desired form. Clearly,

$$(3.2) \quad h_0 \circ (h_1, \dots, h_k) = \Theta\{h_{B(0,1)} \cup C_1^0, \dots, h_{B(0,a_0)} \cup C_{a_0}^0, Q_0\}.$$

For $1 \leq i \leq a_0$, $(h_{B(0,i)} \cup C_i^0)(X_1, \dots, X_k) = \Theta\{X_{B(B(0,i),1)} \cup C_1^{B(0,i)}, X_{B(B(0,i),2)} \cup C_2^{B(0,i)}, \dots, X_{B(B(0,i),a_{B(0,i)})} \cup C_{a_{B(0,i)}}^{B(0,i)}, Q_{B(0,i)}\} \cup C_i^0 = \Theta\{X_{B(B(0,i),1)} \cup (C_1^{B(0,i)} \cup C_i^0), \dots, X_{B(B(0,i),j)} \cup (C_j^{B(0,i)} \cup C_i^0), \dots, X_{B(B(0,i),a_{B(0,i)})} \cup (C_{a_{B(0,i)}}^{B(0,i)} \cup C_i^0), (Q_{B(0,i)} \cup C_i^0)\} = \Theta\{X_{B(i',1)} \cup C_1^{i'}, \dots, X_{B(i',a_{i'})} \cup C_{a_{i'}}^{i'}, Q_{i'}\} = h'_{i'}(X_1, \dots, X_k)$, where $i' = B(0, i)$. We define $\{l_1, l_2, \dots, l_t\}$, $t \leq k$, to be the set of integers such that for each l_s , $1 \leq s \leq t$, there is a term $X_{B(q,p)}$ in $h'_{i'}(X_1, \dots, X_k)$ with $X_{B(q,p)} = X_{l_s}$ for some p, q . For $1 \leq s \leq t$, let $K_{l_s} = \{X_{B(q,p)} \cup C_p^{i'} \mid X_{B(q,p)} = X_{l_s}\}$ and let $K'_{l_s} = \{C_p^{i'} \mid (X_{B(q,p)} \cup C_p^{i'}) \in K_{l_s}\}$. Since $C_p^{i'} \in (\mathcal{U}_v \setminus \mathcal{U}_w)$, each set in K'_{l_s} is disjoint with \mathcal{U}_w . Notice that X_{l_s} is drawn from \mathcal{U}_w . Therefore, (3.2) can be further simplified as follows: $\Theta\{X_{l_1} \cup \Theta\{K'_{l_1}\}, \dots, X_{l_t} \cup \Theta\{K'_{l_t}\}, \Theta\{Q_0, Q'_1, Q'_2, \dots, Q'_{a_0}\}\} = \Theta\{X_{l_1} \cup D_{l_1}, \dots, X_{l_t} \cup D_{l_t}, R\}$, where $D_{l_i} = \Theta\{K'_{l_i}\}$, for $1 \leq i \leq t$ and $R = \Theta\{Q_0, Q'_1, Q'_2, \dots, Q'_{a_0}\}$. It is easy to check that $D_{l_i}, R \in \mathcal{U}_v \setminus \mathcal{U}_w$, and w is a descendant of v . Hence, $h_0 \circ (h_1, \dots, h_k)$ possesses the desired form. \square

We next develop a parallel algorithm for the (r, k, Θ) -subgraph generating problem. For a node x in the current tree H , let $par_H(x)$ ($child_H(x)$) denote the parent

(children) of x and let $sib_H(x)$ denote the *sibling* of x . The subscript H can be omitted if no ambiguity arises. Also let $H(x)$ denote the subtree of H rooted at x . Recall that $R_x = \langle R_{x,1}, \dots, R_{x,k} \rangle$ (respectively, $A_x = \langle A_{x,1}, \dots, A_{x,r} \rangle$) is the list of the target (respectively, auxiliary) subgraphs associated with x .

During the process of executing the tree contraction, we aim at constructing k k -ary functions $h_{x,1}, h_{x,2}, \dots, h_{x,k}$ associated with each node x of the current tree such that each $h_{x,i}$, $1 \leq i \leq k$, possesses the closed form and satisfies the condition described below. Let v be an internal node in the current tree whose left child and right child are u and w , respectively. Let u' be the left child and w' be the right child of v in the original tree. Note that u' and w' are ancestors of u and w in the original tree, respectively. For the remainder of this section, we call u' and w' *replacing ancestors of u and w with respect to v* , respectively. Once $R_{u,i}$ and $R_{w,i}$, $1 \leq i \leq k$, are provided as the inputs of $h_{u,i}$ and $h_{w,i}$, respectively, the target subgraphs of v can be obtained from $Z_{u'} = \langle h_{u,1}(R_{u,1}, \dots, R_{u,k}), \dots, h_{u,k}(R_{u,1}, \dots, R_{u,k}) \rangle \bullet A_{u'}$, and $Z_{w'} = \langle h_{w,1}(R_{w,1}, \dots, R_{w,k}), \dots, h_{w,k}(R_{w,1}, \dots, R_{w,k}) \rangle \bullet A_{w'}$, using the formula

$$(3.3) \quad R_{v,i} = \Theta\{Z_{u',f_{u',i}(1)} \cup Z_{w',g_{w',i}(1)}, Z_{u',f_{u',i}(2)} \cup Z_{w',g_{w',i}(2)}, \dots, Z_{u',f_{u',i}(a_{v,i})} \cup Z_{w',g_{w',i}(a_{v,i})}\}.$$

That is, $R_{u'} = \langle R_{u',1}, \dots, R_{u',k} \rangle = \langle h_{u,1}(R_{u,1}, \dots, R_{u,k}), \dots, h_{u,k}(R_{u,1}, \dots, R_{u,k}) \rangle$ and $R_{w'} = \langle R_{w',1}, \dots, R_{w',k} \rangle = \langle h_{w,1}(R_{w,1}, \dots, R_{w,k}), \dots, h_{w,k}(R_{w,1}, \dots, R_{w,k}) \rangle$. We call the above functions $h_{x,i}$, $1 \leq i \leq k$, computed for each node x in the current tree *the crucial functions of x* . For ease of describing the concept of the crucial function, we demonstrate an example as follows.

Example 1. Consider an internal node v in the original tree T whose left child and right child are u' and w' , respectively. Let u be a proper descendant of u' which is a leaf and let w be a proper descendant of w' (see also Figure 3.2(a)). Initially, the k target subgraphs R_v can be obtained by merging $\langle h_{u',1}(R_{u',1}, \dots, R_{u',k}), \dots, h_{u',k}(R_{u',1}, \dots, R_{u',k}) \rangle \bullet A_{u'}$ and $\langle h_{w',1}(R_{w',1}, \dots, R_{w',k}), \dots, h_{w',k}(R_{w',1}, \dots, R_{w',k}) \rangle \bullet A_{w'}$ in which $R_{u'} = \langle R_{u',1}, \dots, R_{u',k} \rangle$ and $R_{w'} = \langle R_{w',1}, \dots, R_{w',k} \rangle$ are indeterminate. After a sequence of contraction phases, assume T' is the current tree in which the left child and the right child of v are u and w , respectively (see also Figure 3.2(b)). Notice that u' and w' are now replacing ancestors of u and w with respect to v , respectively. R_v are then obtained by merging $\langle h_{u,1}(R_{u,1}, \dots, R_{u,k}), \dots, h_{u,k}(R_{u,1}, \dots, R_{u,k}) \rangle \bullet A_{u'}$ and $\langle h_{w,1}(R_{w,1}, \dots, R_{w,k}), \dots, h_{w,k}(R_{w,1}, \dots, R_{w,k}) \rangle \bullet A_{w'}$. Since $R_{u,i}$ are those subgraphs associated with T before executing the tree contraction, the indeterminate part for generating R_v is reduced to $R_w = \langle R_{w,1}, \dots, R_{w,k} \rangle$. This part is smaller than the previous one.

We next describe the details of our algorithm. Initially, for each node v in the given tree we construct k functions $h_{v,i}(X_1, \dots, X_k) = \Theta\{X_i \cup \emptyset, \emptyset\}, 1 \leq i \leq k$. Clearly, these functions are crucial functions.

In the execution of the tree contraction, assume that *prune*(u) and *bypass*($par(u)$) are performed consecutively. Let $par(u) = v$ and $sib(u) = w$ in the current tree. Let u' and w' be the replacing ancestors of u and w with respect to v , respectively. Assume that $h_{u,i}$ and $h_{w,i}$, $1 \leq i \leq k$, are crucial functions of u and w in the current tree. Thus $R_{u'} = \langle h_{u,1}(R_{u,1}, \dots, R_{u,k}), \dots, h_{u,k}(R_{u,1}, \dots, R_{u,k}) \rangle$ and $R_{w'} = \langle h_{w,1}(R_{w,1}, \dots, R_{w,k}), \dots, h_{w,k}(R_{w,1}, \dots, R_{w,k}) \rangle$. Since u is a leaf, $R_{u,i}$'s are associated with u before executing the tree contraction algorithm. Therefore, the above k target subgraphs $R_{u'}$ can be obtained through function evaluation. On the other hand, since w is not a leaf in the current tree, $R_{w,i}$, $1 \leq i \leq k$, is an

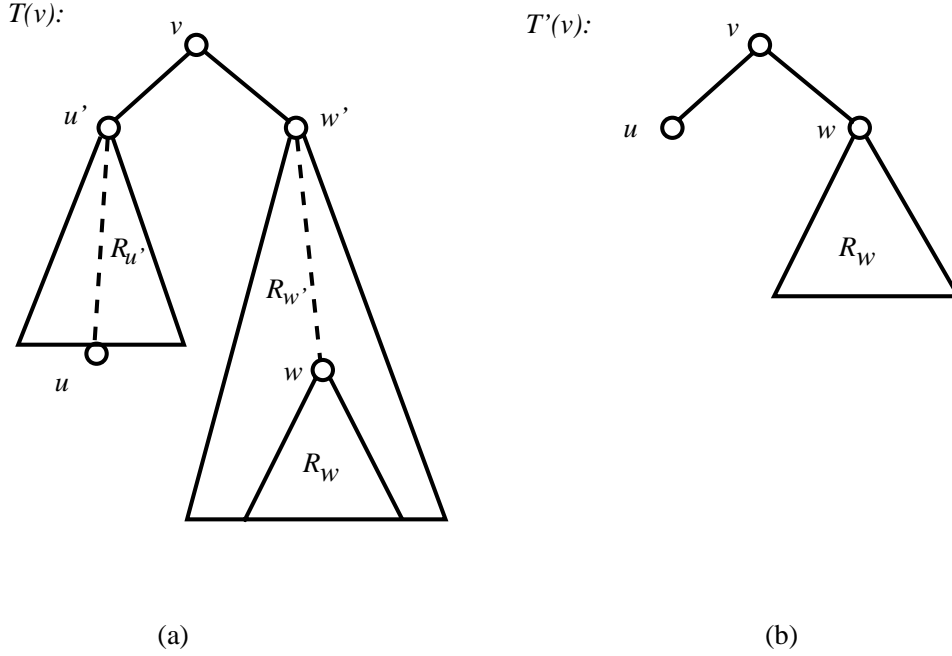


FIG. 3.2. The concept of crucial functions. The indeterminate part for evaluating R_v shown in (a) is smaller than that shown in (b).

indeterminate value represented by variable X_i . Hence, $R_{w'}$ can be represented by $\langle h_{w,1}(X_1, \dots, X_k), \dots, h_{w,k}(X_1, \dots, X_k) \rangle$. By (3.1), we construct k intermediate functions representing the k target subgraphs of v from those of u' and w' by

$$(3.4) \quad R_{v,i} = \Theta\{Z_{u',f_{u',i}(1)} \cup Z_{w',g_{w',i}(1)}, Z_{u',f_{u',i}(2)} \cup Z_{w',g_{w',i}(2)}, \dots, Z_{u',f_{u',i}(a_{v,i})} \cup Z_{w',g_{w',i}(a_{v,i})}\},$$

where $Z_{u',f_{u',i}(j)} = R_{u',f_{u',i}(j)} = h_{u,f_{u',i}(j)}(R_{u,1}, \dots, R_{u,k})$, $Z_{w',g_{w',i}(j)} \in A_{w'}$ if $g_{w',i}(j) > k$, and $Z_{w',g_{w',i}(j)} = h_{w,g_{w',i}(j)}(X_1, \dots, X_k)$ if $g_{w',i}(j) \leq k$.

Similar to the proof of Lemma 3.6, (3.4) can be further simplified as

$$(3.5) \quad R_{v,i} = \Theta\{X_{b_1} \cup C_1, X_{b_2} \cup C_2, \dots, X_{b_a} \cup C_a, Q\},$$

where $b_i \neq b_j$ for two distinct $1 \leq i, j \leq a$, X_{b_i} are variables drawn from \mathcal{U}_w , and $C_i, Q \in (\mathcal{U}_v \setminus \mathcal{U}_w)$.

Therefore, the above functions (constructed after executing $prune(u)$) possess the closed form. Given those functions $R_{v,i}$'s, the contribution to the k target subgraphs of $par(v)$ is obtained by function composition $h_{v,i}(R_{v,1}, \dots, R_{v,k})$ for all $1 \leq i \leq k$. These functions are constructed for w after executing $bypass(par(v))$. By Lemma 3.6, $h_{v,i}(R_{v,1}, \dots, R_{v,k})$, $1 \leq i \leq k$, possesses the closed form. Hence, we have the following lemma.

LEMMA 3.7. *During the process of executing the binary tree contraction on an (r, k, Θ) -subgraph generating tree to remove some nodes, the crucial functions of the remaining nodes of the current tree can be constructed in $O(k^2(r+k))$ time using one processor.*

Proof. This can be shown by induction on the number of contraction phases based on the arguments preceding the lemma. For constructing each of the k functions, there are at most $k(r+k)$ terms generated. These terms can be simplified as the closed form using Θ . Thus the desired complexities follow. \square

THEOREM 3.8. *The (r, k, Θ) -subgraph generating problem can be solved in $O(k^2(r+k) \log n)$ time using $O(n/\log n)$ processors on an EREW PRAM, where n is the number of nodes of the input tree.*

Proof. The algorithm for solving the (r, k, Θ) -subgraph generating problem consists of an initial assignment of k crucial functions to each node of the input tree, and an application of the tree contraction algorithm such that the crucial functions after executing $\text{prune}(v)$ and $\text{bypass}(\text{par}(v))$ are constructed by Lemma 3.7. Once the algorithm terminates, a three-node tree T' results. Let t be the root of T' and y, z be two children of t . Note that the k target subgraphs of y' and z' , the replacing ancestors of y and z with respect to t , can be generated by their corresponding crucial functions. Moreover, the auxiliary subgraphs associated with y' and z' before executing the tree contraction are now maintained in y and z by (3.3). Therefore, according to the operators associated with t , the k target subgraphs of t can be generated. By Lemmas 3.4 and 3.7, the problem can be solved with the stated complexities. \square

DEFINITION 3.9. *Let G be a distance-hereditary graph. A problem \mathcal{P} is said to be an (r, k, Θ) -regular problem on G if \mathcal{P} can be reduced to an (r, k, Θ) -subgraph generating problem \mathcal{B} on a decomposition tree of G such that the following two conditions hold.*

1. *The solution of \mathcal{B} is exactly the solution of \mathcal{P} .*
2. *The reduction scheme takes $O(k^2(r+k) \log n)$ time using $O(n/\log n)$ processors on an EREW PRAM, where n is the number of nodes in the given decomposition tree.*

Note that each (r, k, Θ) -regular problem corresponds to an (r, k, Θ) -subgraph generating tree. This tree is obtained from a decomposition tree \mathcal{D}_G in which some additional data structures are associated with $V(\mathcal{D}_G)$ (refer to Definition 3.1). In the remainder of this section and section 4, we assume that a decomposition tree is given for solving an (r, k, Θ) -regular problem on a distance-hereditary graph. Such a tree will be constructed using a parallel algorithm presented in section 5.

LEMMA 3.10. *Given a decomposition tree of a distance-hereditary graph G , an (r, k, Θ) -regular problem on G can be solved in $O(k^2(r+k) \log n)$ time using $O(n/\log n)$ processors on an EREW PRAM.*

Proof. The proof follows from Definition 3.9 and Theorem 3.8. \square

LEMMA 3.11. *An (r, k, Θ) -regular problem on a distance-hereditary graph can be solved in $O(k(r+k)n + m)$ sequential time.*

Proof. According to Lemma 2.6, a corresponding (r, k, Θ) -subgraph generating tree can be constructed in $O(k(r+k)n + m)$ time. By Lemma 3.3, an (r, k, Θ) -regular problem can be solved within the desired complexity. \square

4. (r, k, Θ) -regular problems. Given a problem \mathcal{P} , a graph G , a subgraph H of G , and a subset S of vertices in H , $\mathcal{P}_S(G, H)$ is a solution to the problem such that this solution has a nonempty intersection with S and is contained in H . For the case of $S = \emptyset$, i.e., $\mathcal{P}_\emptyset(G, H)$, the notation represents a solution to G , and this solution is contained in H . For brevity, let $\mathcal{P}_S(G, G) = \mathcal{P}_S(G)$.

In this section, let $G = (V, E)$ be a distance-hereditary graph and S be the twin set of G . Also let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be distance-hereditary graphs with the twin sets S_1 and S_2 , respectively. Recall that $S = S_1$ if $G = G_1 \oplus G_2$. We will show that the problems demonstrated in sections 4.1–4.5 can be efficiently

parallelized using our strategy. We also note that combining the results of [9, 14], the sequential linear time complexity of these problems can also be obtained.

4.1. The maximum clique problem. A graph is a *clique* if there is an edge between every pair of vertices. We say a clique is in G if it is an induced subgraph of G . We define the *maximum clique problem* \mathcal{C} to be the problem that finds a clique with the maximum number of vertices in the input graph. A previous work to solve this problem on distance-hereditary graph can be found in [19]. Using our notation, we want to solve $\mathcal{C}_\emptyset(G)$. For a primitive distance-hereditary graph $G = (\{v\}, \emptyset)$, $\mathcal{C}_\emptyset(G) = \mathcal{C}_\emptyset(G, G[S]) = \{v\}$.

THEOREM 4.1.

1. In the case of $G = G_1 \otimes G_2$,
 - $\mathcal{C}_\emptyset(G) = \text{MAX}_v\{\mathcal{C}_\emptyset(G_1), \mathcal{C}_\emptyset(G_2), \mathcal{C}_\emptyset(G_1, G_1[S_1]) \cup \mathcal{C}_\emptyset(G_2, G_2[S_2])\}$;
 - $\mathcal{C}_\emptyset(G, G[S]) = \mathcal{C}_\emptyset(G_1, G_1[S_1]) \cup \mathcal{C}_\emptyset(G_2, G_2[S_2])$.
2. In the case of $G = G_1 \oplus G_2$,
 - $\mathcal{C}_\emptyset(G) = \text{MAX}_v\{\mathcal{C}_\emptyset(G_1), \mathcal{C}_\emptyset(G_2), \mathcal{C}_\emptyset(G_1, G_1[S_1]) \cup \mathcal{C}_\emptyset(G_2, G_2[S_2])\}$;
 - $\mathcal{C}_\emptyset(G, G[S]) = \mathcal{C}_\emptyset(G_1, G_1[S_1])$.
3. In the case of $G = G_1 \odot G_2$,
 - $\mathcal{C}_\emptyset(G) = \text{MAX}_v\{\mathcal{C}_\emptyset(G_1), \mathcal{C}_\emptyset(G_2)\}$;
 - $\mathcal{C}_\emptyset(G, G[S]) = \text{MAX}_v\{\mathcal{C}_\emptyset(G_1, G_1[S_1]), \mathcal{C}_\emptyset(G_2, G_2[S_2])\}$.

Proof. The proof is straightforward. \square

For a node v in a decomposition tree \mathcal{D}_G , recall that G_v denote a subgraph of G induced by the leaves of the subtree of \mathcal{D}_G rooted at v . Let S_v denote the twin set of G_v . For convenience, let $V(G_v) = V_v$.

THEOREM 4.2. *The maximum clique problem is a $(1, 2, \text{MAX}_v)$ -regular problem on distance-hereditary graphs.*

Proof. We first reduce the problem to a $(1, 2, \text{MAX}_v)$ -subgraph generating problem. A corresponding $(1, 2, \text{MAX}_v)$ -subgraph generating tree can be constructed by the following steps:

(S1) For each node $v \in V(\mathcal{D}_G)$, set $A_v = \langle \emptyset \rangle$.

(S2) For each internal node v , let u and w be the left child and the right child of v . For $x \in \{u, w\}$, let $Z_x = R_x \bullet A_x = \langle Z_{x,1}, Z_{x,2}, Z_{x,3} \rangle = \langle \mathcal{C}_\emptyset(G_x), \mathcal{C}_\emptyset(G_x, G_x[S_x]), \emptyset \rangle$. Set two integers $a_{v,1}, a_{v,2}$ and construct functions $f_{x,i}$ and $g_{x,i}$, $1 \leq i \leq 2$, according to the following cases:

CASE 1: v is a \otimes -node. Set $a_{v,1} = 3, a_{v,2} = 1$, and $f_{u,1}(1) = g_{w,1}(2) = 1, f_{u,1}(3) = g_{w,1}(3) = f_{u,2}(1) = g_{w,2}(1) = 2, f_{u,1}(2) = g_{w,1}(1) = 3$.

According to Theorem 4.1(1), $\mathcal{C}_\emptyset(G_v) = R_{v,1} = \text{MAX}_v\{Z_{u,f_{u,1}(1)} \cup Z_{w,g_{w,1}(1)}, Z_{u,f_{u,1}(2)} \cup Z_{w,g_{w,1}(2)}, \dots, Z_{u,f_{u,1}(a_{v,1})} \cup Z_{w,g_{w,1}(a_{v,1})}\} = \text{MAX}_v\{Z_{u,1} \cup Z_{w,3}, Z_{u,3} \cup Z_{w,1}, Z_{u,2} \cup Z_{w,2}\}$, and $\mathcal{C}_\emptyset(G_v, G_v[S_v]) = R_{v,2} = \text{MAX}_v\{Z_{u,f_{u,2}(1)} \cup Z_{w,g_{w,2}(1)}, Z_{u,f_{u,2}(2)} \cup Z_{w,g_{w,2}(2)}, \dots, Z_{u,f_{u,2}(a_{v,2})} \cup Z_{w,g_{w,2}(a_{v,2})}\} = \text{MAX}_v\{Z_{u,2} \cup Z_{w,2}\}$.

CASE 2: v is a \oplus -node. Set $a_{v,1} = 3, a_{v,2} = 1$, and $f_{u,1}(1) = g_{w,1}(2) = 1, f_{u,1}(3) = g_{w,1}(3) = f_{u,2}(1) = 2, f_{u,1}(2) = g_{w,1}(1) = g_{w,2}(1) = 3$.

Then, $\mathcal{C}_\emptyset(G_v) = R_{v,1} = \text{MAX}_v\{Z_{u,1} \cup Z_{w,3}, Z_{u,3} \cup Z_{w,1}, Z_{u,2} \cup Z_{w,2}\}$ and $\mathcal{C}_\emptyset(G_v, G_v[S_v]) = R_{v,2} = \text{MAX}_v\{Z_{u,2} \cup Z_{w,3}\}$.

CASE 3: v is a \odot -node. Set $a_{v,1} = 2, a_{v,2} = 2$, and $f_{u,1}(1) = g_{w,1}(2) = 1, f_{u,2}(1) = g_{w,2}(2) = 2, f_{u,1}(2) = g_{w,1}(1) = g_{w,2}(1) = f_{u,2}(2) = 3$.

Then, $\mathcal{C}_\emptyset(G_v) = R_{v,1} = \text{MAX}_v\{Z_{u,1} \cup Z_{w,3}, Z_{u,3} \cup Z_{w,1}\}$ and $\mathcal{C}_\emptyset(G_v, G_v[S_v]) = R_{v,2} = \text{MAX}_v\{Z_{u,2} \cup Z_{w,3}, Z_{u,3} \cup Z_{w,2}\}$.

(S3) For each leaf l corresponding to a primitive distance-hereditary graph $G_l = (\{v\}, \emptyset)$, set two target subgraphs of l to be $R_l = \langle R_{l,1}, R_{l,2} \rangle = \langle \mathcal{C}_\emptyset(G_l), \mathcal{C}_\emptyset(G_l, G_l[S_l]) \rangle$

$= \langle \{v\}, \{v\} \rangle$.

The other two cases for \oplus -node and \odot -node can be verified similarly. Therefore, the maximum clique problem can be reduced to a $(1, 2, \text{MAX}_v)$ -subgraph generating problem. Clearly, steps (S1)–(S3) can be implemented in $O(1)$ time using $O(n)$ processors on an EREW PRAM. As with the aid of Brent's scheduling principle [22], the reduction scheme takes $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM. By Definition 3.9, the theorem holds. \square

4.2. The maximum independent set problem. An *independent set* of a graph is a subset of its vertices such that no two vertices in the subset are adjacent. The *maximum independent set problem* \mathcal{I} is the problem of finding a maximum cardinality independent set in the input graph. A previous sequential linear time algorithm to solve this problem on distance-hereditary graphs can be found in [15]. Using our notation, given an input graph G , a solution is $\mathcal{I}_\emptyset(G)$. For a primitive distance-hereditary graph $G = (\{v\}, \emptyset)$, $\mathcal{I}_\emptyset(G)$ and $\mathcal{I}_S(G)$ are both equal to $\{v\}$, and $\mathcal{I}_\emptyset(G, G[V \setminus S]) = \emptyset$.

THEOREM 4.3.

1. In the case of $G = G_1 \otimes G_2$,
 - $\mathcal{I}_\emptyset(G) = \text{MAX}_v \{ \mathcal{I}_{S_1}(G_1) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2]), \mathcal{I}_{S_2}(G_2) \cup \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]), \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2]) \}$;
 - $\mathcal{I}_S(G) = \text{MAX}_v \{ \mathcal{I}_{S_1}(G_1) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2]), \mathcal{I}_{S_2}(G_2) \cup \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \}$;
 - $\mathcal{I}_\emptyset(G, G[V \setminus S]) = \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2])$.
2. In the case of $G = G_1 \oplus G_2$,
 - $\mathcal{I}_\emptyset(G) = \text{MAX}_v \{ \mathcal{I}_{S_1}(G_1) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2]), \mathcal{I}_{S_2}(G_2) \cup \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]), \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2]) \}$;
 - $\mathcal{I}_S(G) = \mathcal{I}_{S_1}(G_1) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2])$;
 - $\mathcal{I}_\emptyset(G, G[V \setminus S]) = \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{I}_\emptyset(G_2)$.
3. In the case of $G = G_1 \odot G_2$,
 - $\mathcal{I}_\emptyset(G) = \mathcal{I}_\emptyset(G_1) \cup \mathcal{I}_\emptyset(G_2)$;
 - $\mathcal{I}_S(G) = \text{MAX}_v \{ \mathcal{I}_{S_1}(G_1) \cup \mathcal{I}_\emptyset(G_2), \mathcal{I}_{S_2}(G_2) \cup \mathcal{I}_\emptyset(G_1) \}$;
 - $\mathcal{I}_\emptyset(G, G[V \setminus S]) = \mathcal{I}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{I}_\emptyset(G_2, G_2[V_2 \setminus S_2])$.

Proof. The proof is straightforward. \square

As with the proof similar to that of Theorem 4.2, the following result can be obtained.

THEOREM 4.4. *The maximum independent set problem is a $(0, 3, \text{MAX}_v)$ -regular problem on distance-hereditary graphs.*

4.3. The vertex connectivity problem. We now consider the vertex connectivity problem. A *vertex separator* (separator for short) of a graph is a set of vertices whose removal increases the number of connected components or results in a trivial graph, i.e., a graph with no edges. A vertex separator Q of G is *minimal* if any proper subset of Q is not a vertex separator of G . A *minimum vertex separator* of G is a vertex separator with the minimum cardinality. We define the *vertex connectivity problem* \mathcal{V} to be the problem that finds a minimum vertex separator for the input graph. A related work can be found in [26]. Using our notation, a solution on the input graph G is denoted as $\mathcal{V}_\emptyset(G)$.

LEMMA 4.5. *Let Q be a minimal vertex separator of G such that $G = G_1 \otimes G_2$ or $G = G_1 \oplus G_2$. If $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, then there is a connected component H of $G[V \setminus Q]$ such that $V(H) \cap (S_1 \cup S_2) = \emptyset$.*

Proof. Note that G is connected and both $G[V \setminus S_1]$ and $G[V \setminus S_2]$ are disconnected. Thus $S_1 \not\subseteq Q$ and $S_2 \not\subseteq Q$. By assumption, $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, and this lemma holds trivially when $S_1 = Q$ or $S_2 = Q$. We now assume $S_1 \neq Q$ and $S_2 \neq Q$. Hence there is a vertex of S_1 and one of S_2 in $G[V \setminus Q]$. Assume the contrary, that every connected component C of $G[V \setminus Q]$ satisfies $V(C) \cap (S_1 \cup S_2) = \emptyset$. Since every vertex of S_1 is connected to all the vertices of S_2 , $G[V \setminus Q]$ remains connected which contradicts the fact that Q is a vertex separator of G . \square

For a subset X of V , let $N_G(X) = (\bigcup_{v \in X} N_G(v)) \setminus X$. The subscript G in the notations used in this section can be omitted when no ambiguity arises.

LEMMA 4.6. *Let Q be a minimal vertex separator of G . If $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, then $Q \subseteq V(G_i)$ for some $i \in \{1, 2\}$.*

Proof. If $G = G_1 \odot G_2$, the result holds clearly; otherwise, $G = G_1 \otimes G_2$ or $G = G_1 \oplus G_2$. By Lemma 4.5, there exists a connected component H of $G[V \setminus Q]$ such that $V(H) \subseteq V(G_i)$ for some $i \in \{1, 2\}$. Since G is connected and Q is a minimal vertex separator, $N(V(H)) = Q$. By $V(H) \cap (S_1 \cup S_2) = \emptyset$, we know that $N(V(H)) \subseteq V(G_i)$. Therefore, $Q \subseteq V(G_i)$. \square

LEMMA 4.7. *If G is disconnected, then for any connected component C of G , $C \cap S \neq \emptyset$.*

Proof. The proof is straightforward. \square

The following lemma can be shown by the structure characterization of distance-hereditary graphs.

LEMMA 4.8. *Let $G = G_1 \otimes G_2$ or $G = G_1 \oplus G_2$. If $V(G_i) = S_i$ and G_j is disconnected, where $i, j \in \{1, 2\}$ and $i \neq j$, then S_i is a minimal vertex separator of G .*

Let inf be an infinite set of vertices. Given a graph G , let $conn(G)$ be inf if G is connected and be \emptyset if G is disconnected. For a distance-hereditary graph G with the twin set S , a vertex separator Q is called *crucial* if there exists a component H of $G[V \setminus Q]$ such that $V(H) \cap S = \emptyset$. Define $\mathcal{V}_S^2(G)$ to be the problem that finds a minimum crucial vertex separator of G . Let $\mathcal{V}_S^2(G)$ be inf if there is no vertex separator satisfying the requirements. Recall that every connected component of G has a nonempty intersection with S . We define $\mathcal{V}_S^3(G)$ to be the problem that returns inf if $S = V(G)$, and returns $\text{MIN}_v\{V(C) \cap S \mid C \text{ if a connected component of } G \text{ and } (V(C) \setminus S) \neq \emptyset\}$ otherwise. For a primitive distance-hereditary graph $G = (\{v\}, \emptyset)$, $\mathcal{V}_\emptyset(G) = \emptyset$, $\mathcal{V}_S^2(G) = inf$, and $\mathcal{V}_S^3(G) = inf$.

LEMMA 4.9. *Assume that $G = G_1 \otimes G_2$.*

1. *If $S_1 = V(G_1)$ and $S_2 \neq V(G_2)$, then*
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup conn(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\},$
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{\mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}, \text{ and}$
 $\mathcal{V}_S^3(G) = S_1 \cup S_2.$
2. *If $S_2 = V(G_2)$ and $S_1 \neq V(G_1)$, then*
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_2 \cup conn(G_1), S_2 \cup \mathcal{V}_\emptyset(G_1), \mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_1}^3(G_1)\},$
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_1}^3(G_1)\}, \text{ and}$
 $\mathcal{V}_S^3(G) = S_1 \cup S_2.$
3. *If $S_1 = V(G_1)$ and $S_2 = V(G_2)$, then*
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup conn(G_2), S_2 \cup conn(G_1), S_1 \cup \mathcal{V}_\emptyset(G_2), S_2 \cup \mathcal{V}_\emptyset(G_1)\},$
 $\mathcal{V}_S^2(G) = inf, \text{ and}$
 $\mathcal{V}_S^3(G) = inf.$
4. *If $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, then*
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_1}^3(G_1), \mathcal{V}_{S_2}^3(G_2)\},$

$$\begin{aligned} \mathcal{V}_S^2(G) &= \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_1}^3(G_1), \mathcal{V}_{S_2}^3(G_2)\}, \\ \mathcal{V}_S^3(G) &= S_1 \cup S_2. \end{aligned}$$

Proof. We first consider the situation where $S_1 = V(G_1)$ and $S_2 \neq V(G_2)$. Let Q be a minimum vertex separator of G . Note that G is connected. There are five cases.

CASE 1: $Q \subset S_1$. It is impossible because $G[V \setminus Q]$ remains to be connected.

CASE 2: $Q = S_1$. In this case, G_2 must be disconnected. Thus $\mathcal{V}_\emptyset(G)$ equals $S_1 \cup \text{conn}(G_2)$.

CASE 3: $Q \cap S_1 \neq \emptyset$ and $S_1 \setminus Q \neq \emptyset$. Clearly, $Q \cap V(G_2) \neq \emptyset$. There are two subcases.

CASE 3.1: $S_2 \subset Q$. This contradicts the fact that $|Q|$ is the minimum because S_2 is also a vertex separator of G .

CASE 3.2: $S_2 \setminus Q \neq \emptyset$. Thus the vertices in $(S_1 \cup S_2) \setminus Q$ are in the same connected component, say H , of $G[V \setminus Q]$. Let H' be another connected component of $G[V \setminus Q]$. Since G is connected and $V(H') \cap (S_1 \cup S_2) = \emptyset$, $N(V(H')) = Q \subseteq V(G_2)$. This contradicts $Q \cap S_1 \neq \emptyset$.

CASE 4: $S_1 \subseteq Q$ and $Q \cap V(G_2) \neq \emptyset$. In this case, G_2 is connected; otherwise, S_1 is a vertex separator of G . Moreover, for every connected component C of $G[V \setminus Q]$, $V(C) \cap S_2 \neq \emptyset$ (otherwise, $Q \setminus S_1$ is a vertex separator of G). Let $Q' = Q \cap V(G_2)$. Clearly, Q' is a minimal vertex separator of $G_2 = (V_2, E_2)$. We next show Q' is a minimum vertex separator of G_2 . Assume the contrary, that Q'' is a vertex separator of G_2 such that $|Q''| < |Q'|$. There are two situations.

(a) Every connected component of $G_2[V_2 \setminus Q'']$ has a nonempty intersection with S_2 . Clearly, $S_1 \cup Q''$ is a vertex separator of G , and a contradiction arises because $|S_1 \cup Q''| < |S_1 \cup Q'| = |Q|$.

(b) There exists a connected component H of $G_2[V_2 \setminus Q'']$ with $V(H) \cap S_2 = \emptyset$. Then Q'' is a vertex separator of G and $|Q''| < |Q'| < |Q|$ which contradicts the assumption that Q is a minimum separator of G .

By the above discussion, $\mathcal{V}_\emptyset(G)$ equals $S_1 \cup \mathcal{V}_\emptyset(G_2)$.

CASE 5: $Q \cap S_1 = \emptyset$ (i.e., $Q \subseteq V(G_2)$).

CASE 5.1: Q is a vertex separator of G_2 . If every connected component of $G[V \setminus Q]$ has a nonempty intersection with S_2 , then $G[V \setminus Q]$ remains connected. This contradicts the fact that Q is a vertex separator of G . Hence, there exists a connected component H of $G[V \setminus Q]$ such that $V(H) \cap S_2 = \emptyset$. This implies $\mathcal{V}_\emptyset(G) = \mathcal{V}_{S_2}^2(G_2)$.

CASE 5.2: Q is not a vertex separator of G_2 . There exists a connected component H of $G[V \setminus Q]$ such that $V(H) \cap S_1 = \emptyset$ and $V(H) \cap S_2 = \emptyset$. Note that $N(V(H)) \subset V(G_2)$. Moreover, the subgraph induced by $V(H) \cup Q$ is a connected component, say C , of G_2 and $Q = (S_2 \cap V(C))$ by the facts that Q is not a vertex separator of G_2 and $S_2 \cap V(C)$ is a minimal vertex separator of G . This implies that $\mathcal{V}_\emptyset(G) = \mathcal{V}_{S_2}^3(G_2)$.

Combining the above cases, we have $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}$. The equations for computing $\mathcal{V}_S^2(G)$ and $\mathcal{V}_S^3(G)$ can be shown similarly from the structure characterization of G . By Lemmas 4.5–4.8, the other situations can be shown analogously. \square

The following lemma can be shown in a way that is similar to the proof of Lemma 4.9.

LEMMA 4.10. *Assume that $G = G_1 \oplus G_2$.*

1. *If $S_1 = V(G_1)$ and $S_2 \neq V(G_2)$, then*

$$\begin{aligned} \mathcal{V}_\emptyset(G) &= \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}, \\ \mathcal{V}_S^2(G) &= \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}, \text{ and} \end{aligned}$$

- $\mathcal{V}_S^3(G) = S_1$.
- 2. If $S_2 = V(G_2)$ and $S_1 \neq V(G_1)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_2 \cup \text{conn}(G_1), S_2 \cup \mathcal{V}_\emptyset(G_1), \mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_1}^3(G_1)\}$,
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_1}^3(G_1)\}$, and
 $\mathcal{V}_S^3(G) = S_1$.
- 3. If $S_1 = V(G_1)$ and $S_2 = V(G_2)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_2 \cup \text{conn}(G_1), S_1 \cup \mathcal{V}_\emptyset(G_2), S_2 \cup \mathcal{V}_\emptyset(G_1)\}$,
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2)\}$, and
 $\mathcal{V}_S^3(G) = S_1$.
- 4. If $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_1}^3(G_1), \mathcal{V}_{S_2}^3(G_2)\}$,
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_1}^3(G_1), \mathcal{V}_{S_2}^3(G_2)\}$, and
 $\mathcal{V}_S^3(G) = S_1$.

LEMMA 4.11. Assume that $G = G_1 \odot G_2$.

- 1. If $S_1 = V(G_1)$ and $S_2 \neq V(G_2)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_\emptyset(G_1), \mathcal{V}_\emptyset(G_2)\}$,
 $\mathcal{V}_S^2(G) = \mathcal{V}_{S_2}^2(G_2)$, and
 $\mathcal{V}_S^3(G) = \mathcal{V}_{S_2}^3(G_2)$.
- 2. If $S_2 = V(G_2)$ and $S_1 \neq V(G_1)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_\emptyset(G_1), \mathcal{V}_\emptyset(G_2)\}$,
 $\mathcal{V}_S^2(G) = \mathcal{V}_{S_1}^2(G_1)$, and
 $\mathcal{V}_S^3(G) = \mathcal{V}_{S_1}^3(G_1)$.
- 3. If $S_1 = V(G_1)$ and $S_2 = V(G_2)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_\emptyset(G_1), \mathcal{V}_\emptyset(G_2)\}$,
 $\mathcal{V}_S^2(G) = \text{inf}$, and
 $\mathcal{V}_S^3(G) = \text{inf}$.
- 4. If $S_1 \neq V(G_1)$ and $S_2 \neq V(G_2)$, then
 $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{\mathcal{V}_\emptyset(G_1), \mathcal{V}_\emptyset(G_2)\}$,
 $\mathcal{V}_S^2(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^2(G_1), \mathcal{V}_{S_2}^2(G_2)\}$, and
 $\mathcal{V}_S^3(G) = \text{MIN}_v\{\mathcal{V}_{S_1}^3(G_1), \mathcal{V}_{S_2}^3(G_2)\}$.

Proof. The proof follows from the definition of the vertex separator and Lemma 4.6. \square

THEOREM 4.12. The vertex connectivity problem is a $(2, 4, \text{MIN}_v)$ -regular problem on distance-hereditary graphs.

Proof. We first reduce the problem to a $(2, 4, \text{MIN}_v)$ -regular problem. A corresponding $(2, 4, \text{MIN}_v)$ -subgraph generating tree can be constructed by the following steps:

- (S1) For each node $v \in V(\mathcal{D}_G)$, determine whether $S_v = V(G_v)$ and determine whether G_v is connected.
- (S2) For each node $v \in V(\mathcal{D}_G)$, set $A_v = \langle \emptyset, \text{inf} \rangle$.²
- (S3) For each internal node v , let u and w be the left child and the right child of v , respectively. Set four integers $a_{v,1}, \dots, a_{v,4}$ and functions $f_{x,i}$ and $g_{x,i}$, where $x \in \{u, w\}$ and $1 \leq i \leq 4$, according to Lemmas 4.9–4.11. Without loss of generality, assume that v is a \otimes -node. (The case of v being a \oplus - or \ominus -node can be shown similarly.) There are four cases corresponding to 1–4 of Lemma 4.9. Here we consider only that $S_1 = V(G_1)$ and $S_2 \neq V(G_2)$. The other cases are analogous. Let $\mathcal{V}_\emptyset(G_v) = R_{v,1}$, $\mathcal{V}_{S_v}^2(G_v) = R_{v,2}$, $\mathcal{V}_{S_v}^3(G_v) = R_{v,3}$, and $S_v = R_{v,4}$, and let

²It is not difficult to generalize the (r, k, Θ) -subgraph generating tree problem when the input is *inf*.

$Z_v = R_v \bullet A_v = \langle Z_{v,1}, \dots, Z_{v,6} \rangle = \langle \mathcal{V}_\emptyset(G_v), \mathcal{V}_{S_v}^2(G_v), \mathcal{V}_{S_v}^3(G_v), S_v, \emptyset, inf \rangle$. Consider the following two cases.

CASE 1: G_w is connected. In this case, $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\} = \text{MIN}_v\{S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}$ because $\text{conn}(G_2) = inf$. Set $a_{v,1} = 3, a_{v,2} = 2, a_{v,3} = a_{v,4} = 1$, and $g_{w,1}(1) = 1, g_{w,1}(2) = g_{w,2}(1) = 2, g_{w,1}(3) = g_{w,2}(2) = 3, f_{u,1}(1) = f_{u,3}(1) = f_{u,4}(1) = g_{w,3}(1) = g_{w,4}(1) = 4, f_{u,1}(2) = f_{u,1}(3) = f_{u,2}(1) = f_{u,2}(2) = 5$.

According to Lemma 4.9(1), $\mathcal{V}_\emptyset(G_v) = R_{v,1} = \text{MIN}_v\{Z_{u,f_{u,1}(1)} \cup Z_{w,g_{w,1}(1)}, Z_{u,f_{u,1}(2)} \cup Z_{w,g_{w,1}(2)}, \dots, Z_{u,f_{u,1}(a_{v,1})} \cup Z_{w,g_{w,1}(a_{v,1})}\} = \text{MIN}_v\{Z_{u,4} \cup Z_{w,1}, Z_{u,5} \cup Z_{w,2}, Z_{u,5} \cup Z_{w,3}\}$, $\mathcal{V}_{S_v}^2(G_v) = R_{v,2} = \text{MIN}_v\{Z_{u,f_{u,2}(1)} \cup Z_{w,g_{w,2}(1)}, Z_{u,f_{u,2}(2)} \cup Z_{w,g_{w,2}(2)}, \dots, Z_{u,f_{u,2}(a_{v,2})} \cup Z_{w,g_{w,2}(a_{v,2})}\} = \text{MIN}_v\{Z_{u,5} \cup Z_{w,2}, Z_{u,5} \cup Z_{w,3}\}$, $\mathcal{V}_{S_v}^3(G_v) = R_{v,3} = \text{MIN}_v\{Z_{u,f_{u,3}(1)} \cup Z_{w,g_{w,3}(1)}, Z_{u,f_{u,3}(2)} \cup Z_{w,g_{w,3}(2)}, \dots, Z_{u,f_{u,3}(a_{v,3})} \cup Z_{w,g_{w,3}(a_{v,3})}\} = Z_{u,4} \cup Z_{w,4}$, and $S_v = R_{v,4} = Z_{u,4} \cup Z_{w,4}$.

CASE 2: G_w is disconnected. In this case, $\mathcal{V}_\emptyset(G) = \text{MIN}_v\{S_1 \cup \text{conn}(G_2), S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\} = \text{MIN}_v\{S_1, S_1 \cup \mathcal{V}_\emptyset(G_2), \mathcal{V}_{S_2}^2(G_2), \mathcal{V}_{S_2}^3(G_2)\}$ because $\text{conn}(G_2) = \emptyset$. Set $a_{v,1} = 4, a_{v,2} = 2, a_{v,3} = a_{v,4} = 1$, and $g_{w,1}(2) = 1, g_{w,1}(3) = g_{w,2}(1) = 2, g_{w,1}(4) = g_{w,2}(2) = 3, f_{u,1}(1) = f_{u,1}(2) = f_{u,3}(1) = g_{w,3}(1) = f_{u,4}(1) = g_{w,4}(1) = 4, f_{u,1}(4) = f_{u,2}(1) = f_{u,2}(2) = f_{u,3}(1) = g_{w,1}(1) = 5$.

Then, $\mathcal{V}_\emptyset(G_v) = R_{v,1} = \text{MIN}_v\{Z_{u,4} \cup Z_{w,5}, Z_{u,4} \cup Z_{w,1}, Z_{u,5} \cup Z_{w,2}, Z_{u,5} \cup Z_{w,3}\}$, $\mathcal{V}_{S_v}^2(G_v) = R_{v,2} = \text{MIN}_v\{Z_{u,5} \cup Z_{w,2}, Z_{u,5} \cup Z_{w,3}\}$, $\mathcal{V}_{S_v}^3(G_v) = R_{v,3} = \{Z_{u,4} \cup Z_{w,4}\}$, and $S_v = R_{v,4} = Z_{u,4} \cup Z_{w,4}$.

(S4) For each leaf l corresponding to a primitive distance-hereditary graph $(\{v\}, \emptyset)$, let $R_l = \langle R_{l,1}, R_{l,2}, R_{l,3}, R_{l,4} \rangle = \langle \mathcal{V}_\emptyset(G_l), \mathcal{V}_{S_l}^2(G_l), \mathcal{V}_{S_l}^3(G_l), S_l \rangle = \langle \emptyset, inf, inf, \{v\} \rangle$.

Since (S1) can be implemented in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM by utilizing the binary tree contraction and the other steps can be implemented within the desired complexities, the problem is a $(2, 4, \text{MIN}_v)$ -regular problem. \square

4.4. The independent domination problem. We say that in a graph $G = (V, E)$, a subset P of V dominates a subset Q of V if every vertex of Q is either in P or adjacent to a vertex in P . A *dominating set* of a graph $G = (V, E)$ is a subset of V that dominates V . A dominating set is *independent* if the subgraph induced by this set has no edge. The *minimum independent domination problem* \mathcal{ID} is to find a minimum cardinality independent dominating set of the given graph. A previous known sequential result of this problem on distance-hereditary graphs can be found in [6]. Another related work can be found in [5]. For a primitive distance-hereditary graph $G = (\{v\}, \emptyset)$, $\mathcal{ID}_\emptyset(G)$ and $\mathcal{ID}_S(G)$ both equal $\{v\}$, $\mathcal{ID}_\emptyset(G[V \setminus S]) = \emptyset$, and $\mathcal{ID}_\emptyset(G, G[V \setminus S]) = inf$.

THEOREM 4.13.

1. In the case of $G = G_1 \otimes G_2$,
 - $\mathcal{ID}_\emptyset(G) = \text{MIN}_v\{\mathcal{ID}_{S_1}(G_1) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2]), \mathcal{ID}_{S_2}(G_2) \cup \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]), \mathcal{ID}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2, G_2[V_2 \setminus S_2])\}$;
 - $\mathcal{ID}_S(G) = \text{MIN}_v\{\mathcal{ID}_{S_1}(G_1) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2]), \mathcal{ID}_{S_2}(G_2) \cup \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1])\}$;
 - $\mathcal{ID}_\emptyset(G[V \setminus S]) = \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2])$;
 - $\mathcal{ID}_\emptyset(G, G[V \setminus S]) = \mathcal{ID}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2, G_2[V_2 \setminus S_2])$.
2. In the case of $G = G_1 \oplus G_2$,
 - $\mathcal{ID}_\emptyset(G) = \text{MIN}_v\{\mathcal{ID}_{S_1}(G_1) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2]), \mathcal{ID}_{S_2}(G_2) \cup \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]), \mathcal{ID}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2, G_2[V_2 \setminus S_2])\}$;

- $\mathcal{ID}_S(G) = \mathcal{ID}_{S_1}(G_1) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2]);$
- $\mathcal{ID}_\emptyset(G[V \setminus S]) = \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2);$
- $\mathcal{ID}_\emptyset(G, G[V \setminus S]) = \text{MIN}_v\{\mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_{S_2}(G_2), \mathcal{ID}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2, G_2[V_2 \setminus S_2])\}.$

3. In the case of $G = G_1 \odot G_2,$

- $\mathcal{ID}_\emptyset(G) = \mathcal{ID}_\emptyset(G_1) \cup \mathcal{ID}_\emptyset(G_2);$
- $\mathcal{ID}_S(G) = \text{MIN}_v\{\mathcal{ID}_{S_1}(G_1) \cup \mathcal{ID}_\emptyset(G_2), \mathcal{ID}_{S_2}(G_2) \cup \mathcal{ID}_\emptyset(G_1)\};$
- $\mathcal{ID}_\emptyset(G[V \setminus S]) = \mathcal{ID}_\emptyset(G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2[V_2 \setminus S_2]);$
- $\mathcal{ID}_\emptyset(G, G[V \setminus S]) = \mathcal{ID}_\emptyset(G_1, G_1[V_1 \setminus S_1]) \cup \mathcal{ID}_\emptyset(G_2, G_2[V_2 \setminus S_2]).$

As with the method used in the previous problems, we have the following result.

THEOREM 4.14. *The independent domination problem is a $(0, 4, \text{MIN}_v)$ -regular problem on distance-hereditary graphs.*

4.5. The domination problem. The *minimum dominating set problem* \mathcal{D} aims at finding a dominating set in the input graph with the minimum cardinality. A related work on distance-hereditary graph can be found in [5]. For a problem $\mathcal{P}_X(G, H), X = \emptyset$ or $X = S,$ used in this section, we relax the constraint that H is restricted to be a subgraph of $G;$ i.e., the desired dominating set of G is contained in $H,$ and H may not be a subgraph of $G.$ For a primitive distance-hereditary graph $G = (\{v\}, \emptyset), \mathcal{D}_\emptyset(G), \mathcal{D}_S(G)$ and $\mathcal{D}_S(G[V \setminus S], G)$ are all equal to $\{v\},$ and $\mathcal{D}_\emptyset(G[V \setminus S], G) = \emptyset.$

LEMMA 4.15. *Assume that $G = G_1 \otimes G_2.$*

1. If $S_1 = V_1$ and $S_2 \neq V_2,$ then

- $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_{S_2}(G_2), \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2) \cup \{u\}, \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\},$ where $u \in V_1;$
- $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2);$
- $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_2}(G_2), \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2) \cup \{u\}, \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\},$ where $u \in V_1;$
- $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2).$

2. If $S_1 \neq V_1$ and $S_2 = V_2,$ then

- $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \{u\}, \mathcal{D}_\emptyset(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\},$ where $u \in V_2;$
- $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1);$
- $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \{u\}, \mathcal{D}_\emptyset(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\},$ where $u \in V_2;$
- $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1).$

3. If $S_1 = V_1$ and $S_2 = V_2,$ then

- $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1), \mathcal{D}_\emptyset(G_2), \{u, w\}\},$ where $u \in V_1$ and $w \in V_2;$
- $\mathcal{D}_\emptyset(G[V \setminus S], G) = \emptyset;$
- $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1), \mathcal{D}_\emptyset(G_2), \{u, w\}\},$ where $u \in V_1$ and $w \in V_2;$
- $\mathcal{D}_S(G[V \setminus S], G) = \{u\},$ where $u \in V_1 \cup V_2.$

4. If $S_1 \neq V_1$ and $S_2 \neq V_2,$ then

- $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_1}(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_2}(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\};$
- $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2);$
- $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_1}(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_2}(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\};$
- $\mathcal{D}_S(G[V \setminus S], G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_2}(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\}.$

LEMMA 4.16. *Assume that $G = G_1 \oplus G_2.$*

1. If $S_1 = V_1$ and $S_2 \neq V_2$, then
 - $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_{S_2}(G_2), \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2) \cup \{u\}, \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\}$, where $u \in V_1$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_2), \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2) \cup \{u\}\}$, where $u \in V_1$;
 - $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2) \cup \{u\}, \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\}$, where $u \in V_1$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2) \cup \{u\}$, where $u \in V_1$.
2. If $S_1 \neq V_1$ and $S_2 = V_2$, then
 - $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \{w\}, \mathcal{D}_\emptyset(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\}$, where $w \in V_2$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1)\}$;
 - $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \{w\}, \mathcal{D}_{S_1}(G_1)\}$, where $w \in V_2$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1)$.
3. If $S_1 = V_1$ and $S_2 = V_2$, then
 - $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1), \mathcal{D}_\emptyset(G_2), \{u, w\}\}$, where $u \in V_1$ and $w \in V_2$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \{u\}$, where $u \in V_1$;
 - $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1), \{u, w\}\}$, where $u \in V_1$ and $w \in V_2$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \{u\}$, where $u \in V_1$.
4. If $S_1 \neq V_1$ and $S_2 \neq V_2$, then
 - $\mathcal{D}_\emptyset(G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_1}(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_2}(G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\}$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \text{MIN}_v\{\mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2), \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\}$;
 - $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2), \mathcal{D}_{S_1}(G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)\}$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)$.

LEMMA 4.17. Assume that $G = G_1 \odot G_2$.

1. If $S_1 = V_1$ and $S_2 \neq V_2$, then
 - $\mathcal{D}_\emptyset(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)$;
 - $\mathcal{D}_S(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2)$.
2. If $S_1 \neq V_1$ and $S_2 = V_2$, then
 - $\mathcal{D}_\emptyset(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)$;
 - $\mathcal{D}_S(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1)$.
3. If $S_1 = V_1$ and $S_2 = V_2$, then
 - $\mathcal{D}_\emptyset(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \emptyset$;
 - $\mathcal{D}_S(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \{u\}$, where $u \in V_1 \cup V_2$.
4. If $S_1 \neq V_1$ and $S_2 \neq V_2$, then
 - $\mathcal{D}_\emptyset(G) = \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_\emptyset(G_2)$;
 - $\mathcal{D}_\emptyset(G[V \setminus S], G) = \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2)$;
 - $\mathcal{D}_S(G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1) \cup \mathcal{D}_\emptyset(G_2), \mathcal{D}_\emptyset(G_1) \cup \mathcal{D}_{S_2}(G_2)\}$;
 - $\mathcal{D}_S(G[V \setminus S], G) = \text{MIN}_v\{\mathcal{D}_{S_1}(G_1[V_1 \setminus S_1], G_1) \cup \mathcal{D}_\emptyset(G_2[V_2 \setminus S_2], G_2),$

$$\mathcal{D}_{S_2}(G_2[V_2 \setminus S_2], G_2) \cup \mathcal{D}_\emptyset(G_1[V_1 \setminus S_1], G_1)\}.$$

THEOREM 4.18. *The domination problem is a $(2, 4, \text{MIN}_v)$ -regular problem on distance-hereditary graphs.*

Proof. A corresponding $(2, 4, \text{MIN}_v)$ -subgraph generating tree can be constructed by the following steps:

- (S1) For each node $v \in V(\mathcal{D}_G)$, determine whether $S_v = V(G_v)$.
- (S2) For each node $v \in V(\mathcal{D}_G)$, set $A_v = \langle y, \emptyset \rangle$, where $y \in S_v$.
- (S3) For each internal node v , set $a_{v,1}, \dots, a_{v,4}$ and construct corresponding functions, according to Lemmas 4.15–4.17. The details are similar to those in the proofs of Theorems 4.2 and 4.12.
- (S4) For each leaf l corresponding to a primitive distance-hereditary graph $(\{v\}, \emptyset)$, set four target subgraphs of l to be $R_l = \langle \mathcal{D}_\emptyset(G_l), \mathcal{D}_\emptyset(G_l[V_l \setminus S_l], G_l), \mathcal{D}_{S_l}(G_l), \mathcal{D}_{S_l}(G_l[V_l \setminus S_l], G_l) \rangle = \langle \{v\}, \emptyset, \{v\}, \{v\} \rangle$.

Clearly, the above reduction scheme can be implemented with the desired complexities. Therefore, the desired problem is a $(2, 4, \text{MIN}_v)$ -regular problem. \square

5. Parallel constructing a decomposition tree. A parallel algorithm to construct a decomposition tree of a distance-hereditary graph is presented in this section.

5.1. Previously known properties of distance-hereditary graphs. For two arbitrary vertices u and v in a given graph H , let $\text{dist}_H(u, v)$ be the length of a shortest path between u and v in H . Given a vertex u in a connected graph $G = (V, E)$, the *hanging* of G rooted at u , denoted by h_u , is the collection of sets $L_0(u), L_1(u), \dots, L_t(u)$ (or simply L_0, L_1, \dots, L_t without ambiguity), where $t = \max_{v \in V} \text{dist}_G(u, v)$ and $L_i(u) = \{v \in V \mid \text{dist}_G(u, v) = i\}$ for $0 \leq i \leq t$. For any vertex $v \in L_i$ and any vertex set $S \subseteq L_i$, $1 \leq i \leq t$, let $N'(v) = N(v) \cap L_{i-1}$ and $N'(S) = N(S) \cap L_{i-1}$. Any two vertices $x, y \in L_i$ ($1 \leq i \leq t - 1$) are said to be *tied* if x and y have a common neighbor in L_{i+1} .

A vertex subset S is *homogeneous* in a graph $G = (V, E)$ if every vertex in $V \setminus S$ is adjacent to either all or none of the vertices of S . We call a family of subsets *arboreal* if every two subsets of the family are either disjoint or comparable (by set inclusion). For a hanging $h_u = (L_0, L_1, \dots, L_t)$, Hammer and Maffray [15] defined an equivalence relation \equiv_i between vertices of L_i by $x \equiv_i y$, which means x and y are in the same connected component of L_i or x and y are tied. Let \equiv_a be defined on $V(G)$ by $x \equiv_a y$, which means $x \equiv_i y$ for some i .

LEMMA 5.1 (see [2, 11, 15]). *Let h_u be the hanging of G rooted at u and let R_1, R_2, \dots, R_r be the equivalence classes with respect to h_u . Then the following are true.*

1. *For any two vertices x and y in some R_i , $N'(x) = N'(y)$.*
2. *The graph obtained from G by shrinking each R_j into one vertex is a tree rooted at u .*
3. *Each R_j induces a cograph.*
4. *The family $\{N'(R_k) \mid N'(R_k) \subseteq R_i\}$, for $1 \leq i \leq r$, is an arboreal family of homogeneous subsets of $G[R_i]$.*

A hanging of a distance-hereditary graph is depicted in Figure 5.1.

5.2. One-vertex-extension trees of cographs. A graph is *cograph* [8] if it is either a vertex, the complement of a cograph, or the union of two cographs. The cograph is also called the P_4 -free graph which does not contain any induced path of length three [8]. It has been shown that the class of cographs is properly contained in distance-hereditary graphs [15]. A cograph G has a tree representation called *cotree*,

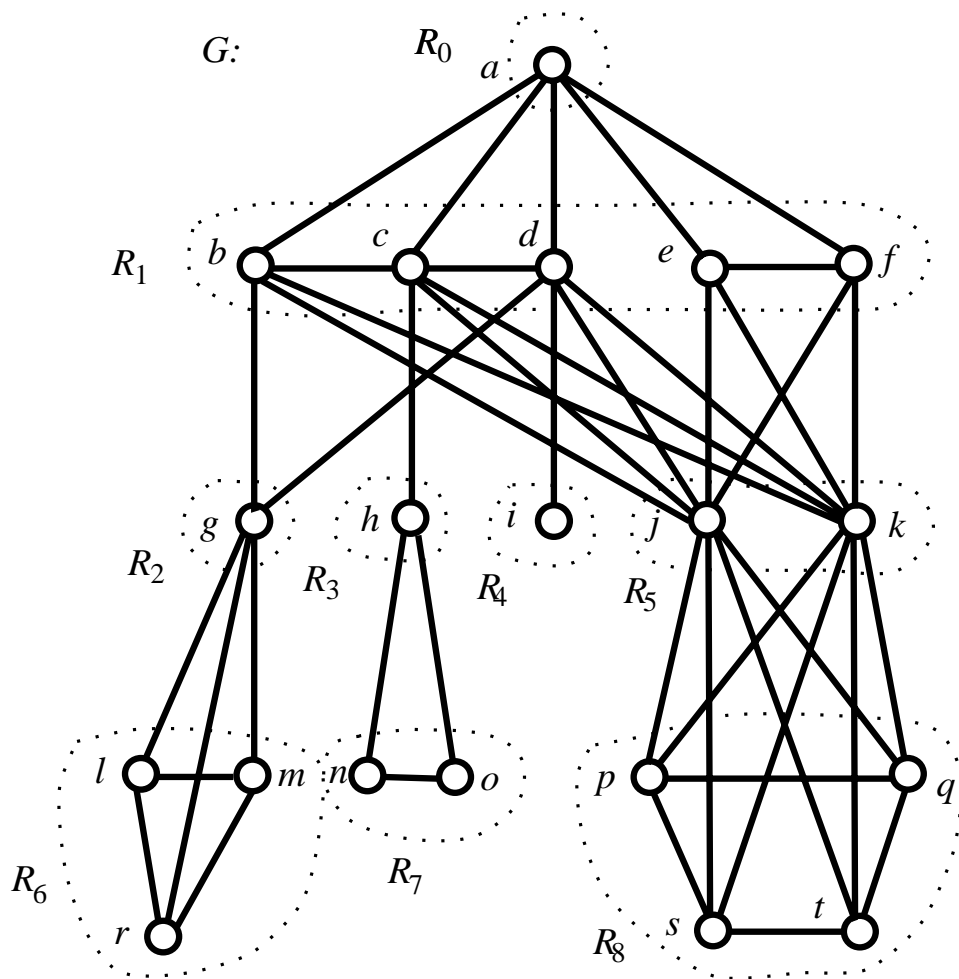


FIG. 5.1. The hanging h_a of a distance-hereditary graph G . The dotted rings depict a partition of $V(G)$ into nine equivalence classes R_0 – R_8 .

denoted by T_G , with the following four properties: (a) the leaves of T_G are the vertices of G ; (b) the internal nodes of T_G are labelled with 0 or 1; (c) 0 nodes and 1 nodes alternate along every path starting from the root; (d) two vertices x and y of G are adjacent if and only if the least common ancestor of x and y in T_G is labelled with 1. Cotrees can be utilized to solve the recognition problem and some other subgraph optimization problems on cographs [17, 23]. Figure 5.2 shows a cograph G and its cotree T_G .

Given a tree T , let $leaf(T)$ be the leaves of T .

LEMMA 5.2. *Let u and v be two leaves in a cotree T_G such that $par(u) = par(v)$. If $par(u)$ is labelled with 1 (respectively, 0), then u and v are true (respectively, false) twins.*

Proof. The proof is straightforward. \square

Given a cograph G represented by its cotree T_G , the graph can be reduced to a single vertex by repeatedly merging twins by the following procedure. We arbitrarily

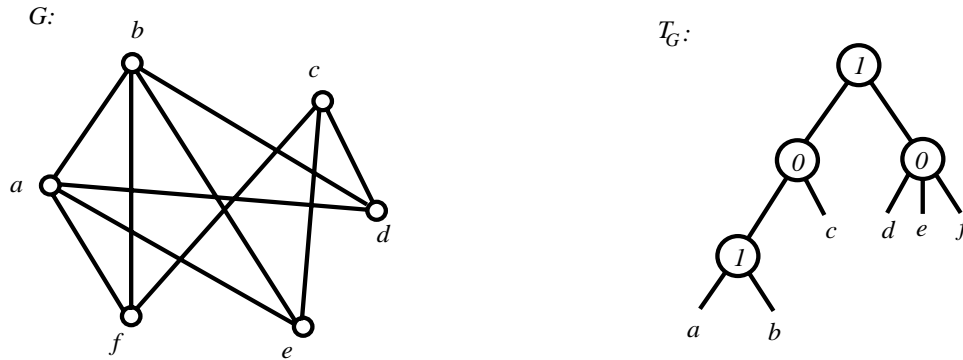


FIG. 5.2. A cograph and its cotree.

find two leaves u and v of the current tree with $par(u) = par(v) = w$. By Lemma 5.2, u and v are twins in the current graph. We delete u from the current graph and the current tree. At the same time, we check whether v is the only child of w in the current tree. If so, we delete w from the current tree and let $par(w)$ be the new parent of v when $w \neq r$. The above procedure is repeatedly executed until the current graph contains only one vertex. Clearly, a one-vertex-extension ordering of G can be obtained by reversing the above process. The above discussion leads to the following algorithm.

Algorithm Tree_1

INPUT: A cograph G .

OUTPUT: A one-vertex-extension tree of G .

- Step 1:** Construct a cotree T_G . Assume that r is the root of T_G .
- Step 2:** Order the leaves of T_G from 1 to $k = |leaf(T_G)|$. Let $order(v)$ be the resulting order associated with $v \in leaf(T_G)$.
- Step 3:** Assign a label to each $u \in T_G$:
Find the vertex $v \in leaf(T_G(u))$ such that $order(v) = \max\{order(w) \mid w \in leaf(T_G(u))\}$. Let $label(u) = v$.
- Step 4:** For each $v \in V(G)$, compute $level(v) = \min\{dist_{T_G}(x, r) \mid label(x) = v\}$.
- Step 5:** Construct a tree \mathcal{E}_G :
 - 5-1.** Let $label(r)$ be the root of \mathcal{E}_G .
 - 5-2.** For each nonroot node $v \in T_G$, let $par(label(v)) = label(par(v))$ if $label(v) \neq label(par(v))$.
 - 5-3.** Label edge $(label(v), label(par(v)))$ as T (respectively, F) if $par(v)$ is a 1 (respectively, 0) node.
- Step 6:** Order the children of each nonleaf vertex $v \in \mathcal{E}_G$:
Assume that v_1, v_2, \dots, v_p are p children of v . Order them by $v_{i_1} < v_{i_2} < \dots < v_{i_p}$ if $level(v_{i_1}) \leq level(v_{i_2}) \leq \dots \leq level(v_{i_p})$, where $1 \leq i_j \leq p$. The resulting tree is a one-vertex-extension tree of G .

An example of executing Algorithm Tree_1 is shown in Figure 5.3. In Figure 5.3(a), the numbers associated with the leaves form an order determined after Step 2. The bold letters associated with internal nodes v represent $label(v)$. In Figure 5.3(b), a one-vertex-extension tree is generated after Steps 4–6.

The correctness follows from the statements preceding the algorithm. The time-processor complexity of Algorithm Tree_1 is analyzed below. In Step 1, T_G can be

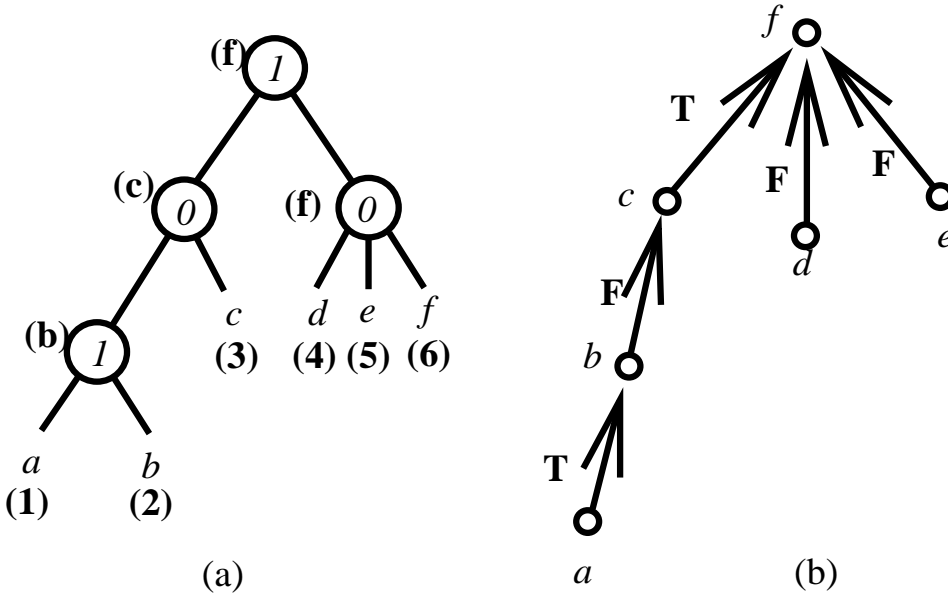


FIG. 5.3. A one-vertex-extension tree shown in (b) is obtained from the given cotree shown in (a).

constructed in $O(\log^2 n)$ time using $O(n + m)$ processors on a CREW PRAM [10]. As with the aid of the Euler-tour, the prefix-sum and the tree contraction techniques [21], Steps 2 and 3 can be implemented in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM. Step 4 can be done within the above complexities using the Euler-tour technique together with the result of finding minimum value [21]. Step 5 can be done in $O(1)$ time using $O(n)$ processors. By utilizing Cole’s parallel merge sort [7], Step 6 can be implemented in $O(\log n)$ time using $O(n)$ processors on an EREW PRAM. Therefore, we have the following theorem.

THEOREM 5.3. *Algorithm Tree.1 correctly constructs a one-vertex-extension tree for a cograph in $O(\log^2 n)$ time using $O(n + m)$ processors on a CREW PRAM.*

5.3. One-vertex-extension trees of distance-hereditary graphs. Throughout this section, G is used to denote a distance-hereditary graph unless stated otherwise.

Let R be an equivalence class of G with respect to a hanging h_u . We call $\Gamma_R = \{Y \subset R \mid \text{there is an equivalence class } R' \text{ with } N'(R') = Y\}$ the *upper neighborhood system in R* and call each $S \in \Gamma_R$, where $S = N'(R')$, the *upper neighborhood of R'* . By Lemma 5.1, Γ_R is an arboreal family of homogeneous subsets of R . We define a partial order \preceq between two different sets Y_p and Y_q in Γ_R with $Y_p \preceq Y_q \Leftrightarrow Y_p \subset Y_q$. According to the partial order \preceq defined on Γ_R , let $\mathcal{U}_R = \{Y_i \mid Y_i \not\subseteq Y_k, \text{ for all } Y_k \in \Gamma_R \text{ and } k \neq i\}$; that is, \mathcal{U}_R is the set of those maximal elements of (\preceq, Γ_R) . We call \mathcal{U}_R the *maximal upper neighborhoods in R* . For a set Y that is the upper neighborhood of some equivalence class, we can also define Γ_Y and \mathcal{U}_Y similarly. In what follows, the notation R is referred to as an equivalence class or an upper neighborhood of some equivalence class if it is not specified.

LEMMA 5.4. *Let $\mathcal{U}_R = \{Q_1, Q_2, \dots, Q_k\}$ and x_i be an arbitrary vertex of Q_i , $1 \leq i \leq k$. The graph $G[(R \setminus \cup_{i=1}^k Q_i) \cup \{x_1, x_2, \dots, x_k\}]$ is a cograph.*

Proof. By the property that every induced subgraph of a P_4 -free graph remains P_4 -free, the result holds. \square

Let $G = (V, E)$ be a cograph and let $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_t\}$ be the set consisting of homogeneous sets of G such that $Q_i \cap Q_j = \emptyset$, $1 \leq i, j \leq t$ and $i \neq j$. Also let $G' = G[(V \setminus (\cup_{i=1}^t Q_i)) \cup \{x_1, x_2, \dots, x_t\}]$, where $x_i \in Q_i$. The following procedure can be used to construct a one-vertex-extension tree of G by merging one-vertex-extension trees of G' and $G[Q_i]$'s.

Procedure 1

- S1:** Construct a one-vertex-extension tree \mathcal{E}' of G' . For each x_i in \mathcal{E}' , $1 \leq i \leq t$, let $(c_1^i, c_2^i, \dots, c_{j_i}^i)$ be the children of x_i .
- S2:** Construct a one-vertex-extension tree \mathcal{E}_i for each $G[Q_i]$. Let r_i be the root of \mathcal{E}_i and let $(d_1^i, d_2^i, \dots, d_{l_i}^i)$ be the children of r_i . Rename the vertex x_i in \mathcal{E}' as r_i .
- S3:** Construct a tree \mathcal{E}_G by identifying each root r_i of \mathcal{E}_i with the vertex r_i in \mathcal{E}' , $1 \leq i \leq t$, such that $(c_1^i, c_2^i, \dots, c_{j_i}^i, d_1^i, d_2^i, \dots, d_{l_i}^i)$ are the resulting children of r_i in \mathcal{E}_G .

LEMMA 5.5. *The tree \mathcal{E}_G constructed in Procedure 1 is a one-vertex-extension tree of a cograph G .*

Proof. We show the lemma by induction on $|\mathcal{Q}| = t$. The base case of $t = 0$ holds clearly. Suppose now that $t > 0$. By the proof of Lemma 5.4, the graph $G_1 = G[(V \setminus Q_1) \cup \{x_1\}]$ is a cograph with $t - 1$ homogeneous sets Q_2, Q_3, \dots, Q_t . By the induction hypothesis, a one-vertex-extension tree \mathcal{E}_{G_1} can be correctly constructed using Procedure 1. Since Q_1 is a homogeneous set, $N_{G_1}(x_1) = (N_G(y) \setminus Q_1)$ for $y \in Q_1 \setminus \{x_1\}$, and $E(G) = E(G_1) \cup E(G[Q_1]) \cup \{(z, b) \mid z \in Q_1, b \in N_{G_1}(x_1)\}$. By executing S2 of Procedure 1, a one-vertex-extension tree \mathcal{E}_1 of Q_1 can be obtained. By S3 of Procedure 1 and the definition of the one-vertex-extension tree, the graph corresponding to \mathcal{E}_G is obtained by connecting $G[Q_1]$ and G_1 through edges $\{(z, b) \mid z \in Q_1, b \in N_{G_1}(x_1)\}$. Hence, \mathcal{E}_G is a one-vertex-extension tree of G . \square

For ordered k children $(v_{i_1}, v_{i_2}, \dots, v_{i_k})$ of a node v_i in \mathcal{E}_G , recall that $\mathcal{E}_G(v_{i_j}, v_i)$ is the subtree of \mathcal{E}_G induced by $v_i, v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}$ and all descendants of $v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}$.

DEFINITION 5.6. *Let R be an equivalence class with respect to a hanging. A one-vertex-extension tree $\mathcal{E}_{G[R]}$ is canonical if for each $Q \in \Gamma_R$ there exist a vertex $v_i \in Q$ and one of its children v_{j_i} such that $\mathcal{E}_{G[R]}(v_{j_i}, v_i)$ is a one-vertex-extension tree of $G[Q]$.*

Given R and Γ_R , the following procedure can be used to construct a canonical one-vertex-extension tree of $G[R]$.

Procedure 2

- S1:** Let $\Gamma_R \cup \{R\} = \{Y_1, Y_2, \dots, Y_t\}$, and let $\mathcal{U}_{Y_i} = \{Y_{i_1}, Y_{i_2}, \dots, Y_{i_{l_i}}\}$, where $Y_1 = R$ and $2 \leq i_j \leq t$ for $1 \leq j \leq l_i$. For each $Y_i \in \Gamma_R$ and $|Y_i| > 1$, select a *shrinking vertex* $y_i \in Y_i$.
- S2:** Let $Y_i' = (Y_i \setminus \cup_{j=1}^{l_i} Y_{i_j}) \cup \{y_{i_1}, y_{i_2}, \dots, y_{i_{l_i}}\}$. Construct a one-vertex-extension trees $\mathcal{E}_{G[Y_i']}$'s, $1 \leq i \leq t$, using Algorithm Tree_1.
- S3:** For each $1 \leq i \leq t$, merge trees $\mathcal{E}_{G[Y_i']}$ and $\mathcal{E}_{G[Y_{i_1}]}, \mathcal{E}_{G[Y_{i_2}]}, \dots, \mathcal{E}_{G[Y_{i_{l_i}}]}$ using Procedure 1.

LEMMA 5.7. *The tree constructed using Procedure 2 is a canonical one-vertex-extension tree for $G[Y_1] = G[R]$.*

Proof. The proof is by induction on $|\Gamma_{Y_1}|$. The base case of $\Gamma_{Y_1} = \emptyset$ trivially holds. Now we consider $|\Gamma_{Y_1}| > 0$. By the induction hypothesis, the canonical one-

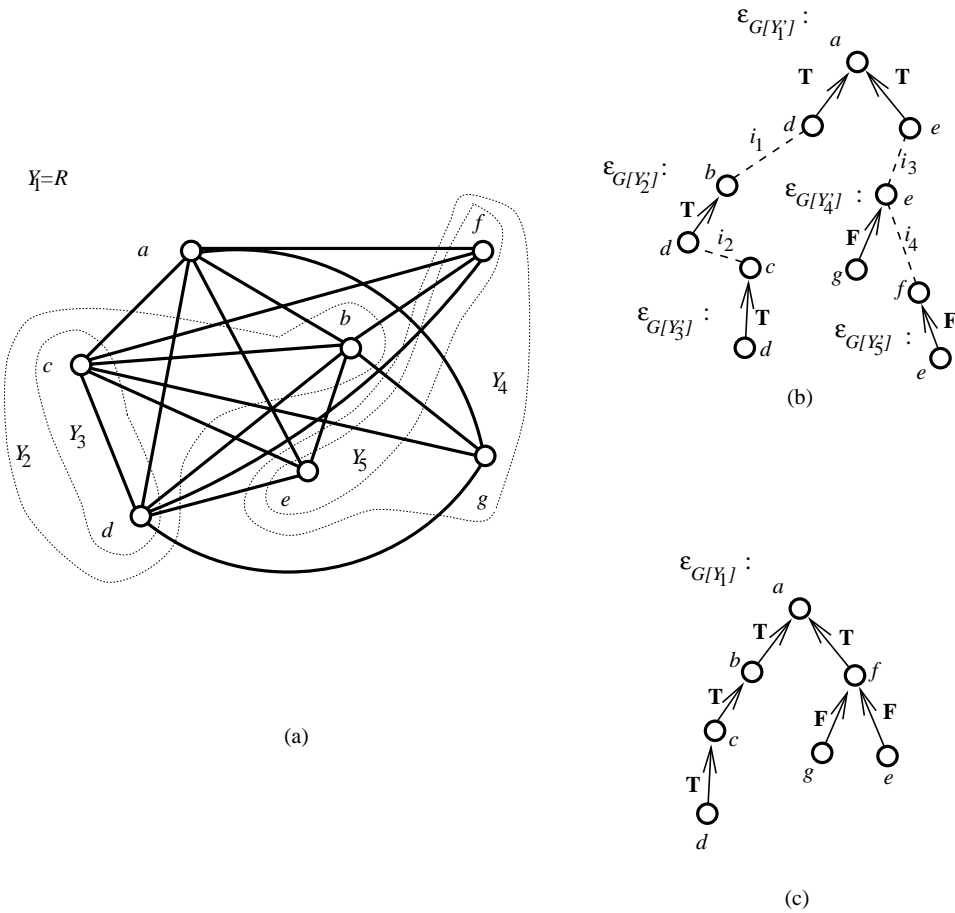


FIG. 5.4. An example of constructing a canonical one-vertex-extension tree using Procedure 2. The dotted lines shown in (b) represent identifying operations.

vertex-extension trees $\mathcal{E}_{G[Y_{1_j}]}$ of $G[Y_{1_j}]$, $1 \leq j \leq l_1$, can be correctly constructed using Procedure 2. By Lemma 5.1(4), Y_{1_j} is a homogeneous set of $G[Y_1]$. By the definition of \mathcal{U}_{Y_1} , $Y_{1_p} \cap Y_{1_q} = \emptyset$, $1 \leq p, q \leq l_1$, and $p \neq q$. The operations used to merge $\mathcal{E}_{G[Y_{1_j}]}$ and $\mathcal{E}_{G[Y_{1_k}]}$'s are based on Procedure 1. Hence, the resulting tree is a one-vertex-extension tree of $G[Y_1]$. Moreover, the canonical property holds from the construction. \square

Figure 5.4 shows an example of generating a canonical one-vertex-extension tree using Procedure 2. Consider $Y_1 = R = \{a, b, c, d, e, f, g\}$, $Y_2 = \{b, c, d\}$, $Y_3 = \{c, d\}$, $Y_4 = \{e, f, g\}$, and $Y_5 = \{e, f\}$ (see Figure 5.4(a)). Note that $\mathcal{U}_{Y_1} = \{Y_2, Y_4\}$, $\mathcal{U}_{Y_2} = \{Y_3\}$, and $\mathcal{U}_{Y_4} = \{Y_5\}$. Let $y_2 = d$, $y_3 = d$, $y_4 = e$, and $y_5 = e$. In Figure 5.4(b), trees $\mathcal{E}_{G[Y_{1_j}]}$ are constructed after S2. The identifying operations shown as dotted lines i_1 – i_4 are then executed in S3. Note that i_1 and i_2 are executed after the labels of d in $\mathcal{E}_{G[Y_{1_1}]}$ and d in $\mathcal{E}_{G[Y_{1_2}]}$ have been changed to b and c , respectively. Also note that the resulting tree can be constructed correctly despite the operations i_3 and i_4 involving the vertex which is the root of $\mathcal{E}_{G[Y_{1_4}]}$ and also the shrinking vertex of Y_5 . Figure 5.4(c) shows the tree produced after executing Procedure 2.

We now present an algorithm to construct a one-vertex-extension tree of a distance-hereditary graph.

Algorithm Tree_2

INPUT: A distance-hereditary graph G .

OUTPUT: A one-vertex-extension tree \mathcal{E}_G .

Step 1: Build a hanging h_u and compute the equivalence classes with respect to h_u .

Step 2: For each equivalence class R , compute Γ_R .

Step 3: For each equivalence class R , generate a canonical one-vertex-extension tree $\mathcal{E}_{G[R]}$ using Procedure 2.

Step 4: For each equivalence class R , let R' be the equivalence class with $N'(R) \subseteq R'$. Find the subtree $\mathcal{E}_{G[N'(R)]}$ in $\mathcal{E}_{G[R']}$, which is a one-vertex-extension tree of $G[N'(R)]$. Let $root(\mathcal{E}_{G[N'(R)]})$ be the root of $\mathcal{E}_{G[N'(R)]}$ and let $(c_1, c_2, \dots, c_{l_{N'(R)}})$ be the children of $root(\mathcal{E}_{G[N'(R)]})$ in $\mathcal{E}_{G[R']}$. Construct \mathcal{E}_G as follows. Let the root of $\mathcal{E}_{G[R]}$ be a new child of $root(\mathcal{E}_{G[N'(R)]})$ which is located between c_{i_R} and c_{i_R+1} for some $1 \leq i_R \leq l_{N'(R)} - 1$ such that $\mathcal{E}_{G[R']}(c_{i_R+1}, root(\mathcal{E}_{G[N'(R)]}))$ equals $\mathcal{E}_{G[N'(R)]}$. The edge $(root(\mathcal{E}_{G[R]}), root(\mathcal{E}_{G[N'(R)]}))$ is labelled with ‘‘P.’’

Figure 5.5 shows the construction of a one-vertex-extension tree of the graph shown in Figure 5.1. The nine canonical one-vertex-extension trees $\mathcal{E}_{G[R_i]}$'s for $0 \leq i \leq 8$ are generated in Step 3. The dotted lines represent those operations executed in Step 4.

Recall that shrinking each equivalence class with respect to the given hanging h_u forms a tree (see Lemma 5.1(2)). We use T_{h_u} to denote such a tree. For each equivalence class R , let ν_R be the node representing R in T_{h_u} . Let $\psi(R) = \{Q \mid \nu_Q \in V(T_{h_u}(\nu_R))\}$ and let $\psi'(R) = \bigcup_{X \in \psi(R)} X$.

LEMMA 5.8. *Algorithm Tree_2 correctly constructs a one-vertex-extension tree of $G[\psi'(R)]$.*

Proof. The proof is by induction on $|\psi(R)|$. The base case of $\psi(R) = \{R\}$ trivially holds. Suppose now that $|\psi(R)| = t > 1$. Let R_1, R_2, \dots, R_r be the equivalence classes with $N'(R_i) \subseteq R$. After Step 3, a canonical one-vertex-extension tree $\mathcal{E}_{G[R]}$ can be constructed. Note that $|\psi(R_i)| < t$ for all $1 \leq i \leq r$. By the induction hypothesis, the one-vertex-extension trees $\mathcal{E}_{G[\psi'(R_i)]}$'s can be correctly constructed using Algorithm Tree_2. After Step 4, the graph corresponding to $\mathcal{E}_{G[\psi'(R)]}$ can be obtained from $G[R]$ (corresponding to $\mathcal{E}_{G[R]}$) and $G[\psi'(R_i)]$ (corresponding to $\mathcal{E}_{G[\psi'(R_i)]}$), $1 \leq i \leq r$, by making R_i and $N'(R_i)$ form a join. According to the structure characterization described in Lemma 5.1, the resulting tree is a one-vertex-extension tree of $G[\psi'(R)]$. \square

By Lemma 5.8, Algorithm Tree_2 correctly constructs a one-vertex-extension tree of $G[\psi'(\{u\})] = G$, where u is the root of the given hanging.

We now analyze the time-processor complexity. Step 1 and Step 2 can be implemented to run in $O(\log^2 n)$ time using $O(n + m)$ processors on a CREW PRAM [19].

To implement Step 3, we need to implement Steps (S1)–(S3) of Procedure 2. In (S1), given $\Gamma_R \cup \{R\} = \{Y_1, Y_2, \dots, Y_t\}$, we find \mathcal{U}_{Y_i} in $O(\log |R|)$ time using $O(\sum_{i=1}^t |Y_i|)$ processors on an EREW PRAM [19]. Clearly, selecting a shrinking vertex can be done in $O(1)$ time using $O(t)$ processors. Thus (S1) can be implemented in $O(\log |R|)$ time using $O(\sum_{i=1}^t |Y_i|)$ processors on an EREW PRAM. The complexities of (S2) are bounded by constructing $\mathcal{E}_{G[Y_i']}$, $1 \leq i \leq t$. By Theorem 5.3, this step can be implemented in $O(\log^2 |R|)$ time using $O(E(G[R]))$ processors on a CREW PRAM. After executing this step, we assume that the children of each node in a given

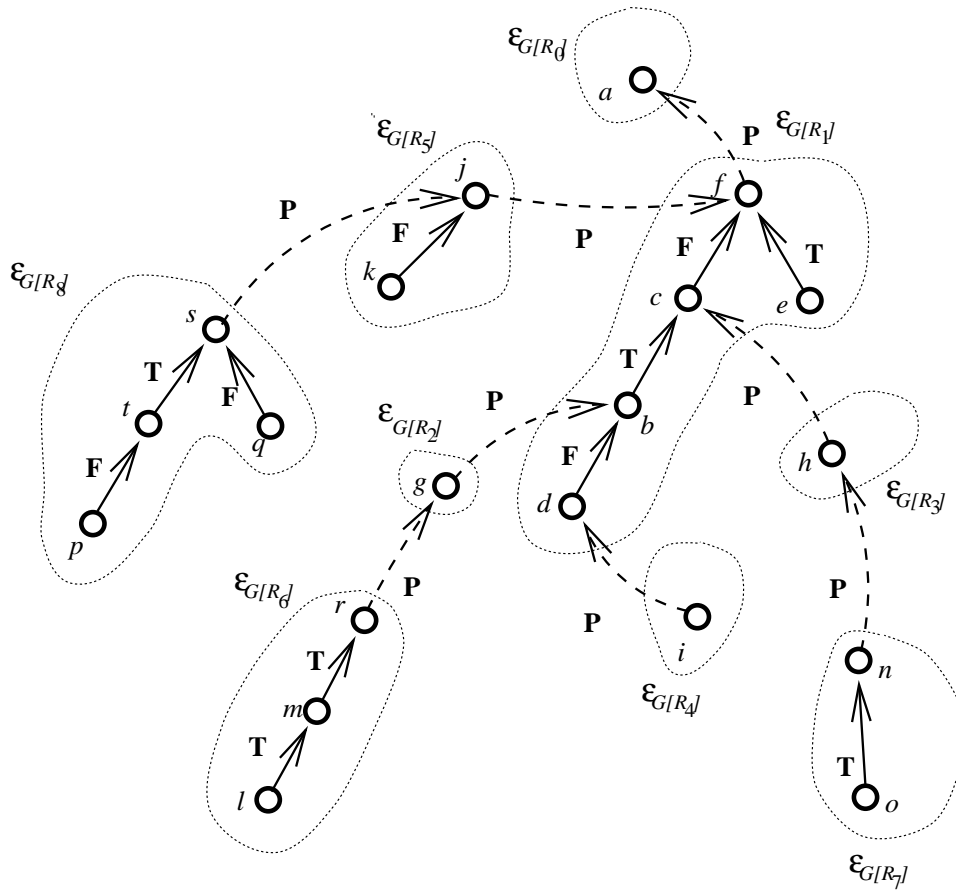


FIG. 5.5. An example of executing Algorithm Tree.2.

tree are manipulated using an ordered list. In (S3), we merge desired trees based on the identifying operations in Procedure 1. Those operations can be implemented in $O(\log t)$ time using $O(t)$ processors on a CREW PRAM. By utilizing the list-ranking technique and the prefix-sum technique [21], we maintain the children of each node in the resulting tree through merging lists. Therefore, Step 3 can be implemented within $O(\log^2 n)$ time using $O(n + m)$ processors on a CREW PRAM. Similarly, Step 4 can be implemented with the desired complexities. Then, we have the following theorem.

THEOREM 5.9. *Algorithm Tree.2 correctly constructs a one-vertex-extension tree of a distance-hereditary graph in $O(\log^2 n)$ time using $O(n+m)$ processors on a CREW PRAM.*

5.4. Decomposition trees of distance-hereditary graphs. Throughout this section, we assume that each vertex of G is represented by its corresponding one-vertex-extension order. By Lemma 2.3, the following recursive method can be used to transform a one-vertex-extension tree into a decomposition tree. Let \mathcal{E} be a given one-vertex-extension tree whose root and leftmost child are x and y , respectively. If (y, x) is labelled with T , then we create a \otimes -node as the root of a decomposition tree $\mathcal{D}_{G[V(\mathcal{E}(x))]}$. If (y, x) is labelled with P , then we create a \oplus -node as the root of

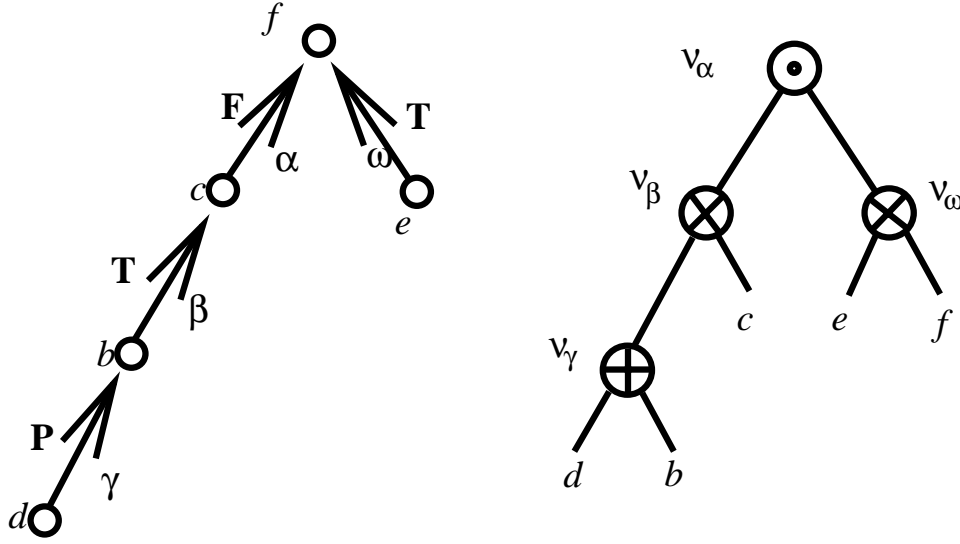


FIG. 5.6. An example of executing Algorithm Tree.3. The tree shown in the left is an input and that shown in the right is its corresponding output.

$\mathcal{D}_{G[V(\mathcal{E}(x))]}$. Otherwise, we create a \odot -node as the root of $\mathcal{D}_{G[V(\mathcal{E}(x))]}$. After recursively constructing $\mathcal{D}_{G[V(\mathcal{E}(y))]}$ and $\mathcal{D}_{G[V(\mathcal{E}(x)) \setminus V(\mathcal{E}(y))]}$, we let the roots of $\mathcal{D}_{G[V(\mathcal{E}(y))]}$ and $\mathcal{D}_{G[V(\mathcal{E}(x)) \setminus V(\mathcal{E}(y))]}$ be the left child and the right child of the created node for (y, x) , respectively. The above method can be implemented using the following non-recursive algorithm.

Algorithm Tree.3

INPUT: A one-vertex-extension tree \mathcal{E}_G .

OUTPUT: A decomposition tree \mathcal{D}_G .

- Step 1:** For each vertex v in \mathcal{E}_G , let $num(v)$ be the one-vertex-extension order associated with v . For each edge $e = (v, par(v))$ in \mathcal{E}_G , let $num(e) = num(v)$.
- Step 2:** For each edge e in \mathcal{E}_G , create an internal node ν_e (\otimes or \oplus or \odot) for \mathcal{D}_G depending on the label of e .
- Step 3:** For each node ν_e , where $e = (v, par(v))$, execute the following operations:
 - (a) If $par(v)$ contains no child w in $V(\mathcal{E}_G)$ such that $num((w, par(v))) > num(e)$, create a node representing $par(v)$ to be the right child of ν_e . Otherwise, find the edge e' next to e . Let the node created for e' be the right child of ν_e .
 - (b) If v is a leaf in $V(\mathcal{E}_G)$, create a node representing v to be the left child of ν_e . Otherwise, find the edge $e' = (z, v)$ such that $num(z) = \min\{num(x) \mid x \in child(v)\}$. Let the node created for e' be the left child of ν_e .

Figure 5.6 shows a one-vertex-extension tree with its corresponding decomposition tree. The nodes $\nu_\alpha, \nu_\beta, \nu_\gamma, \nu_\omega$ are created in Step 2 of Algorithm Tree.3. The left child and the right child of each node ν_e , where e is an edge in $\{\alpha, \beta, \gamma, \omega\}$, are determined in Step 3 of Algorithm Tree.3.

The correctness follows from the statements preceding the algorithm. Based on the data structure maintained in Algorithm `Tree_2` and the Euler-tour technique [21], we have the following result.

THEOREM 5.10. *Algorithm `Tree_3` correctly transforms a one-vertex-extension tree into a decomposition tree in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM.*

6. Discussion and conclusion. In this paper, we first define the (r, k, Θ) -subgraph generating problem on trees. We solve this problem in $O((rk + k^2)n)$ sequential time, and in $O(k^2(r + k) \log n)$ time using $O(n/\log n)$ processors on an EREW PRAM, where n is the number of nodes of the given tree. We then develop a general problem-solving paradigm used to reduce a class of subgraph optimization problems on distance-hereditary graphs to its corresponding (r, k, Θ) -subgraph generating problems. Using this paradigm, we define a class of (r, k, Θ) -regular problems on distance-hereditary graphs. Let $T_d(|V|, |E|)$ and $P_d(|V|, |E|)$ denote the time complexity and processor complexity required to construct a decomposition tree of a distance-hereditary graph $G = (V, E)$ on a PRAM model M_d . We show that an (r, k, Θ) -regular problem on a distance-hereditary graph $G = (V, E)$ can be solved in sequential $O((rk + k^2)n + m)$ time, and in $O(T_d(n, m) + \log n)$ time using $O(P_d(n, m) + n/\log n)$ processors on M_d . We also show that $T_d(n, m) = O(\log^2 n)$, $P_d(n, m) = O(n + m)$ under a CREW PRAM.

Several fundamental graph problems are shown to be (r, k, Θ) -regular, including the maximum clique problem, the maximum independent set problem, the vertex connectivity problem, the domination problem, and the independent domination problem. Therefore, the above problems can be solved in linear time, and in $O(\log^2 n)$ time using $O(n + m)$ processors on a CREW PRAM. Opposed to less parallel results on distance-hereditary graphs, our method classifies a class of problems on distance-hereditary graphs to be in NC. We believe that more graph problems can be shown to be in (r, k, Θ) -regular class.

We note that Golumbic and Rotics [14] showed that a distance-hereditary graph has clique-width at most three and can be represented by a so called 3-expression. Using this structure, it is shown that a class of problems can be solved in sequential linear time on distance-hereditary graphs if those problems can be represented in monadic second order logic with quantification over vertex sets only (MSOL problems for short) [9]. Note that Bodlaender and Hagerup [4] developed a general parallel algorithm to solve several subgraph optimization problems on special classes of graphs with bounded tree-width. However, the tree-width of distance-hereditary graphs is not bounded. It is hopeful and certainly interesting to see if clique-width can be used similarly to solve subgraph optimization problems in parallel. However, to the best of our knowledge, no such result exists.

In [24], Miller and Teng presented a systemic method for the design of efficient parallel algorithms for the dynamic evaluation of computation trees and/or expressions. Their method involves the use of uniform closure properties of certain classes of unary functions. In this paper, we extend their work by considering k -ary functions. Let D be the power set of some given set and let MIN (respectively, MAX) be the operator defined on a subset of D that returns a set with the minimum (respectively, maximum) cardinality. We show that a class algebraic computation tree over $\{D, \text{MIN}, \text{MAX}, \cup\}$ can be optimally evaluated using a class of k -ary functions which is closed under the composition.

Acknowledgments. The authors are deeply appreciative for the comments and suggestions given by the editor and the two anonymous referees.

REFERENCES

- [1] K. ABRAHAMSON, N. DADOUN, D. G. KIRKPATRICK, AND T. PRZYTYCKA, *A simple parallel tree contraction algorithm*, J. Algorithms, 10 (1989), pp. 287–302.
- [2] H. J. BANDELT AND H. M. MULDER, *Distance-hereditary graphs*, J. Combin. Theory Ser. B, 41 (1986), pp. 182–208.
- [3] C. BERGE, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.
- [4] H. L. BODLAENDER AND T. HAGERUP, *Parallel algorithms with optimal speedup for bounded treewidth*, SIAM J. Comput., 27 (1998), pp. 1725–1746.
- [5] A. BRANDSTÄDT AND F. F. DRAGAN, *A linear time algorithm for connected γ -domination and Steiner tree on distance-hereditary graphs*, Networks, 31 (1998), pp. 177–182.
- [6] M. S. CHANG, S. Y. HSIEH, AND G. H. CHEN, *Dynamic programming on distance-hereditary graphs*, in Proceedings of the 7th International Symposium on Algorithms and Computation (ISAAC'97), Lecture Notes in Comput. Sci. 1350, Springer-Verlag, Berlin, 1997, pp. 344–353.
- [7] R. COLE, *Parallel merge sort*, SIAM J. Comput., 17 (1988), pp. 770–785.
- [8] D. G. CORNEIL, H. LERCHS, AND L. S. BURLINGHAM, *Complement reducible graphs*, Discrete Appl. Math., 3 (1981), pp. 163–174.
- [9] B. COURCELLE, J. A. MAKOWSKY, AND U. ROTICS, *Linear time solvable optimization problems on graphs of bounded clique-width*, Theory Comput. Syst., 33 (2000), pp. 125–150.
- [10] E. DAHLHAUS, *Efficient parallel recognition algorithms of cographs and distance-hereditary graphs*, Discrete Appl. Math., 57 (1995), pp. 29–44.
- [11] A. D'ATRI AND M. MOSCARINI, *Distance-hereditary graphs, Steiner trees, and connected domination*, SIAM J. Comput., 17 (1988), pp. 521–538.
- [12] F. F. DRAGAN, *Dominating cliques in distance-hereditary graphs*, in Algorithm Theory-SWAT'94: 4th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 824, Springer-Verlag, Berlin, 1994, pp. 370–381.
- [13] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [14] M. C. GOLUMBIC AND U. ROTICS, *On the clique-width of perfect graph classes*, in Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99), Lecture Notes in Comput. Sci. 1665, Springer-Verlag, Berlin, 1999, pp. 135–147.
- [15] P. L. HAMMER AND F. MAFFRAY, *Complete separable graphs*, Discrete Appl. Math., 27 (1990), pp. 85–99.
- [16] X. HE, *Efficient parallel algorithms for solving some tree problems*, in Proceedings of the 24th Allerton Conference on Communication, Control, and Computing, 1986, pp. 777–786.
- [17] X. HE, *Parallel algorithm for cograph recognition with applications*, J. Algorithms, 15 (1993), pp. 284–313.
- [18] E. HOWORKA, *A characterization of distance-hereditary graphs*, Quart. J. Math. Oxford Ser. (2), 28 (1977), pp. 417–420.
- [19] S.-Y. HSIEH, C. W. HO, T.-S. HSU, M. T. KO, AND G. H. CHEN, *Efficient parallel algorithms on distance-hereditary graphs*, Parallel Process. Lett., 9 (1999), pp. 43–52.
- [20] S.-Y. HSIEH, C. W. HO, T.-S. HSU, M. T. KO, AND G. H. CHEN, *A faster implementation of a parallel tree contraction scheme and its application on distance-hereditary graphs*, J. Algorithms, 35 (2000), pp. 50–81.
- [21] J. JA'JA', *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA, 1992.
- [22] R. M. KARP AND V. RAMACHANDRAN, *Parallel algorithms for shared memory machines*, in Handbook of Theoretical Computer Science, North-Holland, Amsterdam, 1990, pp. 869–941.
- [23] R. LIN AND S. OLARIU, *An optimal parallel matching algorithm for cographs*, J. Parallel Distrib. Comput., 22 (1994), pp. 26–36.
- [24] G. L. MILLER AND S. H. TENG, *Tree-based parallel algorithm design*, Algorithmica, 19 (1997), pp. 369–389.
- [25] F. NICOLAI, *Hamiltonian Problems on Distance-Hereditary Graphs*, Technique report, Gerhard-Mercator University, Duisburg, Germany, 1994.
- [26] Y. SHILOACH AND U. VISKIN, *An $O(\log n)$ parallel connectivity algorithm*, J. Algorithms, 3 (1982), pp. 57–63.

- [27] H. G. YEH AND G. J. CHANG, *Weighted connected domination and Steiner trees in distance-hereditary graphs*, *Discrete Appl. Math.*, 87 (1998), pp. 245–253.
- [28] H. G. YEH AND G. J. CHANG, *Linear-Time Algorithms for Bipartite Distance-Hereditary Graphs*, manuscript.
- [29] H. G. YEH AND G. J. CHANG, *The path-partition problem in bipartite distance-hereditary graphs*, *Taiwanese J. Math.*, 2 (1998), pp. 353–360.

DOMINATION IN GRAPHS APPLIED TO ELECTRIC POWER NETWORKS*

TERESA W. HAYNES[†], SANDRA M. HEDETNIEMI[‡], STEPHEN T. HEDETNIEMI[‡], AND
MICHAEL A. HENNING[§]

Abstract. The problem of monitoring an electric power system by placing as few measurement devices in the system as possible is closely related to the well-known vertex covering and dominating set problems in graphs. We consider the graph theoretical representation of this problem as a variation of the dominating set problem and define a set S to be a power dominating set of a graph if every vertex and every edge in the system is monitored by the set S (following a set of rules for power system monitoring). The minimum cardinality of a power dominating set of a graph G is the power domination number $\gamma_P(G)$. We show that the power dominating set (PDS) problem is NP-complete even when restricted to bipartite graphs or chordal graphs. On the other hand, we give a linear algorithm to solve the PDS for trees. In addition, we investigate theoretical properties of $\gamma_P(T)$ in trees T .

Key words. domination, power domination, electric power monitoring

AMS subject classification. 05C69

PII. S0895480100375831

1. Introduction. Electric power companies need to continually monitor their system's state as defined by a set of state variables (for example, the voltage magnitude at loads and the machine phase angle at generators [6]). One method of monitoring these variables is to place phase measurement units (PMUs) at selected locations in the system. Because of the high cost of a PMU, it is desirable to minimize their number while maintaining the ability to monitor (observe) the entire system. A system is said to be *observed* if all of the state variables of the system can be determined from a set of measurements (e.g., voltages and currents).

Let $G = (V, E)$ be a graph representing an electric power system, where a vertex represents an electrical node (a substation bus where transmission lines, loads, and generators are connected) and an edge represents a transmission line joining two electrical nodes. The problem of locating a smallest set of PMUs to monitor the entire system is a graph theory problem closely related to the well-known vertex covering and domination problems. Hence, this problem is not only of interest in the power system industry but also as a new problem in graph theory. For a thorough study of domination and related subset problems as well as terminology not defined here, we refer the reader to two books [4, 5].

A PMU measures the state variable (voltage and phase angle) for the vertex at which it is placed and its incident edges and their endvertices. (These vertices and edges are said to be *observed*.) The other observation rules are as follows:

*Received by the editors July 24, 2000; accepted for publication (in revised form) June 20, 2002; published electronically September 10, 2002. This research was supported in part by the South African National Research Foundation and the University of Natal, South Africa.

<http://www.siam.org/journals/sidma/15-4/37583.html>

[†]Department of Mathematics, East Tennessee State University, Johnson City, TN 37614 (haynes@etsu.edu).

[‡]Department of Computer Science, Clemson University, Clemson, SC 29634 (shedet@cs.clemson.edu, hedet@cs.clemson.edu).

[§]Department of Mathematics, University of Natal, Private Bag X01, Pietermaritzburg, 3209 South Africa (henning@math.unp.ac.za).

1. Any vertex that is incident to an observed edge is observed.
2. Any edge joining two observed vertices is observed.
3. If a vertex is incident to a total of $k > 1$ edges and if $k - 1$ of these edges are observed, then all k of these edges are observed.

For a given set of vertices P representing the nodes where the PMUs are placed, the following algorithm determines the sets of (observed) vertices C and edges F .

Let $P \subseteq V$ be the set of vertices where the PMUs are placed.

1. Initialize $C = P$ and $F = \{e \in E(G) \mid e \text{ is incident to a vertex in } P\}$.
2. Add to C any vertex not already in C which is incident to an edge in F .
3. Add to F any edge not already in F such that
 - a. both of its endvertices are in C or
 - b. it is incident to a vertex v of degree greater than one for which all the other edges incident to v are in F .
4. If steps 2 and 3 fail to locate any new edges or vertices for inclusion, stop. Otherwise, go to step 2.

Therefore, to solve the power system monitoring problem, we want $C = V(G)$ and $F = E(G)$, and we want to minimize $|P|$. This monitoring problem was introduced and studied in [1, 2, 3, 6].

A set $S \in V(G)$ is a *dominating set* in a graph $G = (V, E)$ if every vertex in $V - S$ has at least one neighbor in S . The cardinality of a minimum dominating set of G is the *domination number* $\gamma(G)$. Considering the power system monitoring problem as a variation of the dominating set problem, we define a set S to be a *power dominating set* (PDS) if every vertex and every edge in G is observed by S . The *power domination number* $\gamma_P(G)$ is the minimum cardinality of a power dominating set of G . A dominating set (respectively, power dominating set) of G with minimum cardinality is called a $\gamma(G)$ -set (respectively, $\gamma_P(G)$ -set).

Since any dominating set is a PDS, we have the following observation.

OBSERVATION 1. For any graph G , $1 \leq \gamma_P(G) \leq \gamma(G)$.

Obviously, any graph G with $\gamma(G) = 1$ demonstrates sharpness of both the upper and lower bounds. Our next observation gives examples of graphs having a power domination number equal to 1.

OBSERVATION 2. For the graph G , where $G \in \{K_n, C_n, P_n, K_{2,n}\}$, $\gamma_P(G) = 1$.

The *corona* of two graphs G and H , denoted $G \circ H$, is the graph formed from one copy of G and $|V(G)|$ copies of H where the i th vertex of G is adjacent to every vertex in the i th copy of H . For another example, the corona $G = P_k \circ K_2$ has $\gamma_P(G) = \gamma(G) = k$. On the other hand, the difference $\gamma(G) - \gamma_P(G)$ can be arbitrarily large. For instance, the corona of a star $T = K_{1,k} \circ K_1$ has $\gamma_P(T) = 1 < k + 1 = \gamma(T)$ for $k \geq 1$.

We note that every graph H is the induced subgraph of a graph G having $\gamma_P(G) = \gamma(G)$. Consider, for example, the corona $G = (H \circ \overline{K}_2)$, where $\gamma_P(G) = \gamma(G)$. Hence, we have the following observation.

OBSERVATION 3. There is no forbidden subgraph characterization of the graphs G for which $\gamma_P(G) = \gamma(G)$.

Suppose that G is a graph with maximum degree at least 3 and that S is a $\gamma_P(G)$ -set that contains a vertex v of degree less than 3. Let u be a vertex of degree at least 3 at minimum distance from v in G . Then, $(S - \{v\}) \cup \{u\}$ is also a minimum power dominating set of G . Hence, our next observation follows immediately.

OBSERVATION 4. If G is a graph with maximum degree at least 3, then G contains a $\gamma_P(G)$ -set in which every vertex has degree at least 3.

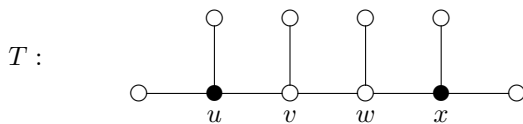


FIG. 1. A tree T with $\gamma_P(T) = 2$.

We note, however, that the vertices of small degree play a significant role in determining the power domination number of a graph. In particular, it is not necessarily true that if G' is the graph obtained from a graph G by subdividing one edge of G , then $\gamma_P(G') = \gamma_P(G)$.

For example, if T is the tree shown in Figure 1 and if T' is the tree obtained from T by subdividing the edge vw once, then $\gamma_P(T') > \gamma_P(T)$ and $\{u, x\}$ is a $\gamma_P(T)$ -set, while $\{u, v, x\}$ is a $\gamma_P(T')$ -set.

Boisen, Baldwin, and Mili [2] investigated approximation algorithms to find a solution to the power system monitoring problem. In this paper, we show that the PDS problem is NP-complete even when restricted to bipartite or chordal graphs, give a linear time algorithm to find a PDS in trees, and study theoretical properties of the power domination number in trees.

2. NP-completeness. In this section we show that the following decision problem is NP-complete even when restricted to bipartite or chordal graphs.

POWER DOMINATING SET (PDS)

INSTANCE: A graph $G = (V, E)$ and a positive integer $k > 1$.

QUESTION: Does G have a PDS of size at most k ?

A relatively simple modification of the standard proof of NP-completeness of DOMINATING SET, given in [4], suffices to establish the NP-completeness of the PDS even when restricted to bipartite graphs.

THEOREM 5. *POWER DOMINATING SET is NP-complete for bipartite graphs.*

Proof. We first show that $PDS \in \mathcal{NP}$. This is easy to do since one can verify a “yes” instance of PDS in polynomial time. That is, for a graph $G = (V, E)$, a positive integer k , and an arbitrary subset $S \subseteq V$ with $|S| \leq k$, it is easy to verify in polynomial time whether S is a PDS.

We next construct a reduction from the well-known NP-complete problem 3-SAT.

3-SAT

INSTANCE: A set $U = \{u_1, u_2, \dots, u_n\}$ of *variables* and a set $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ of 3-element sets, called *clauses*, where each clause C_i contains three distinct occurrences of either a variable u_i or its complement \bar{u}_i .

QUESTION: Does \mathcal{C} have a satisfying truth assignment, i.e., an assignment of TRUE and FALSE to the variables in U such that at least one variable (or its complement) in each clause $C_i \in \mathcal{C}$ is assigned the value TRUE?

Given an instance \mathcal{C} of 3-SAT, we construct an instance $G(\mathcal{C})$ of the PDS as follows. For each variable u_i , construct a cycle on four vertices C_4 , where two non-adjacent vertices are labelled u_i and \bar{u}_i . These cycles and vertices are called *variable cycles* and *variable vertices*, respectively. For each clause $C_j = \{u_i, u_k, u_l\}$ create two nonadjacent vertices labelled $C_{j,1}$ and $C_{j,2}$ (called *clause vertices*), and add edges: $(u_i, C_{j,1}), (u_i, C_{j,2}), (u_k, C_{j,1}), (u_k, C_{j,2}), (u_l, C_{j,1}), (u_l, C_{j,2})$. (For an example, see Figure 2.) By construction, the graph $G(\mathcal{C})$ is bipartite.

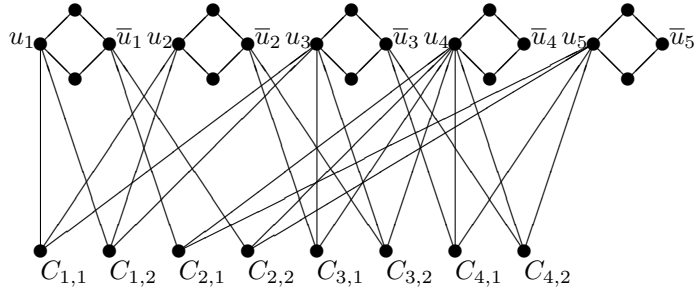


FIG. 2. An instance of 3-SAT where $U = \{u_1, u_2, u_3, u_4, u_5\}$ and $\mathcal{C} = \{\{u_1, u_2, u_3\}, \{\bar{u}_1, u_4, u_5\}, \{\bar{u}_2, u_3, u_4\}, \{\bar{u}_3, u_4, u_5\}\}$.

We show that \mathcal{C} has a satisfying truth assignment if and only if the graph $G(\mathcal{C})$ has a PDS of size at most $k = n$.

Suppose first that \mathcal{C} has a satisfying truth assignment. We create a PDS S in $G(\mathcal{C})$ as follows: If variable u_i is assigned the value TRUE, then put variable vertex u_i in S ; otherwise, put variable vertex \bar{u}_i in S . The set S is a PDS for two reasons: (i) One variable vertex in each variable cycle C_4 belongs to S and therefore both vertices of degree 2 in the variable cycle are dominated. Thus, every edge and vertex in each variable cycle C_4 is observed. (ii) Since the set S corresponds to a truth assignment for \mathcal{C} , every clause vertex C_j is dominated by at least one vertex in S . Therefore, all edges between a variable vertex and a clause vertex, $C_{j,1}$ or $C_{j,2}$, are observed, and S is a PDS of size at most $k = n$.

Conversely, we must show that if $G(\mathcal{C})$ has a PDS of size at most $k = n$, then \mathcal{C} has a satisfying truth assignment. Notice first that if S is a PDS of $G(\mathcal{C})$, then it must contain at least one vertex from each variable cycle C_4 . This follows from the observation that no set of clause vertices can suffice to dominate or observe all four vertices of a variable cycle. Therefore, $|S| \geq n$, i.e., $|S| = n$.

Notice next that a set S , all of whose vertices lie in the variable cycles (in fact, one from each C_4), can be a PDS if and only if every clause vertex is dominated by at least one vertex in S . Although it is true that vertices in the variable cycles, which are not variable vertices, can dominate both of their corresponding variable vertices, it is not possible for these dominated variable vertices to subsequently observe any clause vertex, since there are two (unobserved) edges from any given variable vertex to clause vertices $C_{j,1}$ and $C_{j,2}$. \square

A similar transformation using the graph in Figure 3 with additional edges such that the variable vertices induce a complete subgraph yields the result for chordal graphs, which we state without proof.

THEOREM 6. *POWER DOMINATING SET is NP-complete for chordal graphs.*

3. Trees. In this section, we investigate the power domination number of a tree. For this purpose, we shall need the following notation. For any vertex $v \in V$, the *open neighborhood* of v , denoted by $N(v)$, is the set $\{u \in V \mid uv \in E\}$ and its *closed neighborhood* $N[v] = N(v) \cup \{v\}$. For a set $S \subseteq V$, its *open neighborhood* $N(S) = \cup_{v \in S} N(v)$ and its *closed neighborhood* $N[S] = N(S) \cup S$. The *private neighbor set* of a vertex v with respect to a set S , denoted $pn[v, S]$, is the set $N[v] - N[S - \{v\}]$. If $pn[v, S] \neq \emptyset$ for some vertex v and some $S \subseteq V$, then every vertex of $pn[v, S]$ is called

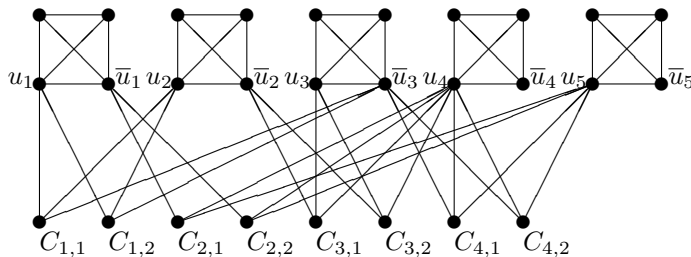


FIG. 3. Instance of 3-SAT where $U = \{u_1, u_2, u_3, u_4, u_5\}$ and $C = \{\{u_1, u_2, u_3\}, \{\bar{u}_1, u_4, u_5\}, \{\bar{u}_2, u_3, u_4\}, \{\bar{u}_3, u_4, u_5\}\}$.

a private neighbor of v with respect to S , or just an S -pn.

If T is a tree rooted at r and v is a vertex of T , then the *level number* of v , which we denote by $\ell(v)$, is the length of the unique r - v path in T . If a vertex u of T is adjacent to v and $\ell(u) > \ell(v)$, then u is called a *child* of v , and v is the *parent* of u , written $v = \text{parent}(u)$. A vertex w is a *descendant* of v (and v is an *ancestor* of w) if the level numbers of the vertices on the v - w path are monotonically increasing. We let $D(v)$ denote the set of descendants of v , and we define $D[v] = D(v) \cup \{v\}$. The *maximal subtree of T rooted at v* is the subtree of T induced by $D[v]$ and is denoted by T_v . We will refer to an endvertex of T as a *leaf*. A vertex adjacent to a leaf is called a *support vertex*, and a vertex adjacent to two or more leaves is called a *strong support vertex*.

Let T be the tree formed from a star by subdividing any number of its edges any number of times; that is, T has at most one vertex of degree 3 or more. We call such a tree T a *spider*. A path, for example, is a special case of a spider.

THEOREM 7. For any tree T , $\gamma_P(T) = 1$ if and only if T is a spider.

Proof. Suppose T is a spider. If T is a path, then any vertex of T forms a PDS in T . On the other hand, if T is not a path, then the vertex of maximum degree in T forms a PDS in T . In any event, $\gamma_P(T) = 1$. This proves the sufficiency.

To prove the necessity, suppose that T is not a spider. Then T contains at least two vertices, say u and v , of degree at least 3. We may assume that T is rooted at v . Let S be any PDS of T . If $|S| = 1$, then, renaming u and v if necessary, we may assume that no vertex in the maximal subtree T_u rooted at u belongs to S . However then no edge in T_u is observed, a contradiction. Therefore, $|S| \geq 2$. Hence, $\gamma_P(T) \geq 2$. \square

Next we characterize those trees T with equal domination and power domination numbers. We will use the following observation.

OBSERVATION 8. If v is a strong support vertex in a graph G , then v is in every $\gamma(G)$ -set and every $\gamma_P(G)$ -set.

For a set S and a vertex $v \in S$, let A_v denote the set $V - (S \cup \text{pn}[v, S])$.

THEOREM 9. For any tree T of order at least 3, $\gamma_P(T) = \gamma(T)$ if and only if T has a unique $\gamma(T)$ -set S and every vertex in S is a strong support vertex.

Proof. Since $1 \leq \gamma_P(T) \leq \gamma(T)$, obviously the theorem holds if $\gamma(T) = 1$. Thus assume that $\gamma(T) \geq 2$. Let T be a tree with a unique $\gamma(T)$ -set S , where every vertex in S is a strong support vertex. Then Observation 8 implies that S is also a unique $\gamma_P(T)$ -set. Hence, $\gamma_P(T) = \gamma(T)$.

For the necessity, assume that $\gamma_P(T) = \gamma(T)$. Suppose to the contrary that $v \in S$ for some $\gamma(T)$ -set S and that v is not a strong support vertex.

Assume that $pn[v, S] = \{v\}$. Then every neighbor of v is dominated by $S - \{v\}$ and v is an isolate in the induced subgraph $\langle S \rangle$. Furthermore, all vertices and edges in $T - v$ are observed by $S - \{v\}$. Since T is a nontrivial tree, v has at least one neighbor, say u , in $V - S$. Moreover, every edge incident to u in $T - v$ is observed by $S - \{v\}$ implying that, in T , uv is observed by $S - \{v\}$ for all $u \in N(v)$. Then v is also observed by $S - \{v\}$, and so $S - \{v\}$ is a PDS for T , contradicting the fact that $\gamma_P(T) = \gamma(T)$. Hence, $|pn[v, S] - \{v\}| \geq 1$.

Suppose there is no endvertex in $pn[v, S]$. Then each vertex in $pn[v, S] - \{v\}$ has a neighbor in A_v . Moreover, since T is tree, $|N[u] \cap (pn[v, S] \cup \{v\})| \leq 1$ for all $u \in V - S$. Since $S - \{v\}$ dominates A_v , it follows that all the vertices and edges of $\langle A_v \rangle$ are observed by $S - \{v\}$. Furthermore, in T each edge between a vertex in $pn[v, S] \cup \{v\}$ and a vertex in A_v is incident to a vertex where all its other incident edges are observed. Thus, every edge between $pn[v, S] \cup \{v\}$ and A_v is observed by $S - \{v\}$ implying that $pn[v, S] - \{v\}$ is observed by $S - \{v\}$. Now, for each edge uv where $u \in pn[v, S]$, all other edges incident to u are observed by $S - \{v\}$, and so uv is also observed. Any other edge incident to v must be incident to a vertex in $S - \{v\}$ and hence is observed. Thus, v is also observed and $S - \{v\}$ is a PDS of T with cardinality less than $\gamma_P(T)$, again a contradiction. Hence, $pn[v, S]$ contains an endvertex, say u (adjacent to v).

By our assumption that no vertex is a strong support vertex, every vertex in $pn[v, S] - \{u\}$ has a neighbor in A_v and is observed by $S - \{v\}$ as before. Using the same argument as before, we can show that the vertices $V - \{u\}$ and every edge except uv is observed by $S - \{v\}$, so uv is also observed. Hence, u is observed and again we have a PDS with cardinality less than $\gamma_P(T)$, a contradiction. We conclude that every vertex in S is a strong support vertex and S is a unique dominating set of T as claimed. \square

Before proceeding further, we define the *spider number* of a tree T , denoted by $sp(T)$, to be the minimum number of subsets into which $V(T)$ can be partitioned so that each subset induces a spider. We call such a partition a *spider partition* and each set of the partition a *spider subset*.

LEMMA 10. For any tree T , $sp(T) \leq \gamma_P(T)$.

Proof. We proceed by induction on $m = \gamma_P(T)$. Suppose $m = 1$. Then, by Theorem 7, T is a spider, and so $sp(T) = 1$. Suppose, then, that all trees T' with $\gamma_P(T') = m$, where $m \geq 1$, satisfy $sp(T') \leq \gamma_P(T')$. Let T be a tree with $\gamma_P(T) = m + 1$. Let $S = \{v_1, v_2, \dots, v_{m+1}\}$ be a $\gamma_P(T)$ -set. By Observation 4, we may assume that each vertex of S has degree at least 3 in T .

Let T be rooted at the vertex v_{m+1} . Relabeling if necessary, we may assume that v_1 is the vertex of S at maximum distance from v_{m+1} in T . Then v_1 has at least two children, and each descendant of v_1 has degree at most 2 in T . Let u_1 be the ancestor of v_1 of degree at least 3 that is at a minimum distance from v_1 . Then either u_1 is the parent of v_1 or every internal vertex on the u_1-v_1 path has degree 2 in T . We consider two possibilities, depending on whether $u_1 \in S$.

Case 1. $u_1 \in S$.

Let w_1 be the child of u_1 on the u_1-v_1 path (possibly, $w_1 = v_1$), and let T_1 be the maximal subtree rooted at w_1 . Then, T_1 is a spider with v_1 as the vertex of degree exceeding 2. Let T_2 be the component of $T - u_1w_1$ containing u_1 , i.e., $T_2 = T - D[w_1]$. Then, $S - \{v_1\}$ is a PDS of T_2 , while $\{v_1\}$ is a PDS of T_1 . In particular, this implies that

$\gamma_P(T_2) \leq m$ and that T_1 is a spider. If $\gamma_P(T_2) < m$, then adding $\{v_1\}$ to a $\gamma_P(T_2)$ -set produces a PDS of T of cardinality less than $m + 1 = \gamma_P(T)$, which is impossible. Hence, $\gamma_P(T_2) \geq m$. Consequently, $\gamma_P(T_2) = m$. Applying the inductive hypothesis to T_2 , $\text{sp}(T_2) \leq \gamma_P(T_2) = m$. Adding the subset $V(T_1)$ to a partition of $V(T_2)$ into $\text{sp}(T_2)$ spider subsets produces a partition of $V(T)$ into $\text{sp}(T_2) + 1 \leq m + 1 = \gamma_P(T)$ spider subsets. Thus, $\text{sp}(T) \leq \gamma_P(T)$.

Case 2. $u_1 \notin S$.

Let w_1, w_2, \dots, w_k be the children of u_1 , where w_1 is the child on u_1 on the u_1 - v_1 path (possibly, $v_1 = w_1$). For $i = 1, 2, \dots, k$, let $W_i = D[w_i]$. In particular, $v_1 \in W_1$. Since $u_1 \notin S$ and since S is a PDS of T , all except possibly one of the sets W_2, \dots, W_k contains a vertex of S .

Suppose one of the sets W_2, \dots, W_k contains no vertex of S . We may assume that $W_2 \cap S = \emptyset$. Necessarily, W_2 induces a path (possibly trivial). If $k \geq 3$, then $|W_i \cap S| \geq 1$ for $i = 3, \dots, k$. Let T_1 be the tree induced by $W_1 \cup W_2 \cup \{u_1\}$. Then, T_1 is a spider with v_1 as the vertex of degree exceeding 2. Let $T_2 = T - D[u_1]$. If $k \geq 3$, let T_i be the maximal subtree rooted at w_i for $i = 3, \dots, k$. For $i = 1, 2, \dots, k$, let $S_i = S \cap V(T_i)$. Note that $S_1 = \{v_1\}$. Let $i \in \{1, 2, \dots, k\}$. Then, S_i is a PDS of T_i . In particular, this implies that $\gamma_P(T_i) \leq |S_i|$. If $\gamma_P(T_i) < |S_i|$, then adding $S - S_i$ to a $\gamma_P(T_i)$ -set produces a PDS of T of cardinality less than $m + 1 = \gamma_P(T)$, which is impossible. Hence, $\gamma_P(T_i) \geq |S_i|$. Consequently, $\gamma_P(T_i) = |S_i| \leq m$. Applying the inductive hypothesis to T_i , $\text{sp}(T_i) \leq \gamma_P(T_i) = |S_i|$. Hence, there exists a spider partition of $V(T_i)$ of cardinality $\text{sp}(T_i)$. Thus, $V(T)$ can be partitioned into $\text{sp}(T_1) + \text{sp}(T_2) + \dots + \text{sp}(T_k) \leq |S_1| + |S_2| + \dots + |S_k| = |S| = \gamma_P(T)$ spider subsets. Thus, $\text{sp}(T) \leq \gamma_P(T)$.

Suppose, on the other hand, that each of the sets W_2, \dots, W_k contains a vertex of S . Then, $|W_i \cap S| \geq 1$ for $i = 1, 2, \dots, k$. Let T_1 be the tree induced by $W_1 \cup \{u_1\}$. Then, T_1 is a spider with v_1 as the vertex of degree exceeding 2. For $i = 2, \dots, k$, let T_i be the maximal subtree rooted at w_i . Let $T_{k+1} = T - D[u_1]$. For $i = 1, 2, \dots, k + 1$, let $S_i = S \cap V(T_i)$. Then, proceeding as in the previous paragraph, each S_i is a $\gamma_P(T_i)$ -set and, applying the inductive hypothesis to T_i , $V(T)$ can be partitioned into $\text{sp}(T_1) + \text{sp}(T_2) + \dots + \text{sp}(T_{k+1}) \leq |S_1| + |S_2| + \dots + |S_{k+1}| = |S| = \gamma_P(T)$ spider subsets. Thus, once again, $\text{sp}(T) \leq \gamma_P(T)$. \square

LEMMA 11. *For any tree T , $\gamma_P(T) \leq \text{sp}(T)$.*

Proof. We proceed by induction on $m = \text{sp}(T)$. Suppose $m = 1$. Then T is a spider, and so, by Theorem 7, $\gamma_P(T) = 1 = \text{sp}(T)$. Suppose, then, that all trees T' with $\text{sp}(T') = m$, where $m \geq 1$, satisfy $\gamma_P(T') \leq \text{sp}(T')$. Let T be a tree with $\text{sp}(T) = m + 1$. Let $\{V_1, V_2, \dots, V_{m+1}\}$ be a spider partition of $V(T)$. For $i = 1, 2, \dots, m + 1$, let T_i be the tree induced by V_i ; that is, $T_i = \langle V_i \rangle$.

Let G be the graph with vertex set $\{V_1, V_2, \dots, V_{m+1}\}$ where two vertices V_i and V_j are adjacent in G if and only if there is an edge of T joining a vertex of V_i and a vertex of V_j . Since T is a tree, so too is G . Without loss of generality, we may assume that V_1 is an endvertex of G and that V_1 and V_2 are adjacent in G . Hence, there exist vertices u_1 and u_2 in V_1 and V_2 , respectively, such that u_1u_2 is an edge of T , and this is the only edge joining a vertex of V_1 and a vertex of $V(T) - V_1$. Let $T' = T - V_1$; that is, T' is obtained from T by deleting the vertices in the subset V_1 . If $\text{sp}(T') < m$, then we can add the subset V_1 to a minimum spider partition of $V(T')$ to produce a spider partition of $V(T)$ of cardinality $\text{sp}(T') + 1 < m + 1 = \text{sp}(T)$, which is impossible. Hence, $\text{sp}(T') \geq m$. Since $\{V_2, \dots, V_{m+1}\}$ is a spider partition of $V(T')$, $\text{sp}(T') \leq m$. Consequently, $\text{sp}(T') = m$. Applying the inductive hypothesis

to T' , $\gamma_P(T') = m$.

Let S' be a $\gamma_P(T')$ -set. Thus all vertices and edges of T' are observed by S' . Let v_1 be a vertex of maximum degree in the spider T_1 . Let $S = S' \cup \{v_1\}$. Then the vertex u_1 is observed from the vertex v_1 , while the vertex u_2 is observed from a vertex of S' . Thus the edge u_1u_2 is observed from S . It follows that S is a PDS of T , and so $\gamma_P(T) \leq m + 1 = \text{sp}(T)$. \square

As an immediate consequence of Lemmas 10 and 11, we have the following result which states that the power domination number of a tree is precisely the spider number of the tree.

THEOREM 12. *For any tree T , $\gamma_P(T) = \text{sp}(T)$.*

As a consequence of Theorem 12, we can determine a lower bound on the power domination of a tree in terms of the number of vertices of degree at least 3.

THEOREM 13. *If T is a tree having k vertices of degree at least 3, then*

$$\gamma_P(T) \geq \frac{k+2}{3},$$

and this bound is sharp.

Proof. Suppose $\gamma_P(T) = m$. Then, by Theorem 12, $\text{sp}(T) = m$. Let $\{V_1, V_2, \dots, V_m\}$ be a spider partition of $V(T)$. For $i = 1, 2, \dots, m$, let T_i be the tree induced by V_i ; that is, $T_i = \langle V_i \rangle$. Since each T_i is a spider, there is at most one vertex of degree at least 3 in T_i .

Let G be the graph with vertex set $\{V_1, V_2, \dots, V_m\}$ as defined in the proof of Theorem 12. Then G is a tree with $m - 1$ edges. Each vertex of T that is not incident with any of these $m - 1$ edges of G and that is not a vertex of degree at least 3 in the spider T_i to which it belongs has degree at most 2 in T . It follows that T contains at most $m + 2(m - 1) = 3m - 2$ vertices of degree at least 3. Thus, $k \leq 3m - 2$, or, equivalently, $\gamma_P(T) = m \geq (k + 2)/3$.

That this bound is sharp may be seen as follows. Let T be the corona of a path on $3n$ vertices; that is, $T = P_{3n} \circ K_1$. (The corona of a path is also called a *comb*.) Let the path be denoted by v_1, v_2, \dots, v_{3n} . Then, $\cup_{i=1}^n \{v_{3i-1}\}$ is a PDS of G of cardinality n , and so $\gamma_P(T) \leq n$. However, since T has $k = 3n - 2$ vertices of degree at least 3 in T , it follows that $\gamma_P(T) \geq (k + 2)/3 = n$. Consequently, $\gamma_P(T) = (k + 2)/3 = n$. \square

We show next that the power domination of a tree is at most a third of the order of the tree.

THEOREM 14. *For any tree T of order $n \geq 3$, $\gamma_P(T) \leq n/3$ with equality if and only if T is the corona $T' \circ \overline{K_2}$, where T' is any tree.*

Proof. We proceed by induction on the number m of vertices of degree at least 3 in a tree T of order $n \geq 3$. If $m = 0$ or $m = 1$, then T is a spider, and so, by Theorem 7, $\gamma_P(T) = 1 \leq n/3$. Assume, then, that for all trees T' of order $n' \geq 3$ with fewer than m vertices of degree at least 3, where $m \geq 2$, that $\gamma_P(T') \leq n'/3$. Let T be a tree of order n with m vertices of degree at least 3.

We may assume that T is rooted at the vertex r . Let v be a vertex of degree at least 3 at maximum distance from r in T . Then every descendant of v in T has degree at most 2, and so the maximal subtree of T rooted at v is a spider. Let T' be the tree of order n' obtained from T by removing v and all its descendants. Since $m \geq 2$, $n' \geq 3$. Furthermore, since v has at least two children, $n' \leq n - 3$. By construction, T' has fewer than m vertices of degree at least 3. Applying the inductive hypothesis to T' , there exists a PDS S' of T' satisfying $|S'| \leq n'/3$. Since $S \cup \{v\}$ is a PDS of T , $\gamma_P(T) \leq |S| + 1 \leq n'/3 + 1 \leq n/3$.

Obviously, if T is the corona $T' \circ \overline{K}_2$, where T' is a tree, $\gamma_P(T) = |V(T')| = n/3$. Suppose that T is a tree of order $n \geq 3$ satisfying $\gamma_P(T) = n/3$. Among all minimum spider partitions of T , let $P = \{V_1, V_2, \dots, V_{n/3}\}$ be chosen to contain the minimum number of sets of cardinality at most 2. For $i = 1, \dots, n/3$, we may assume that v_i is a vertex of maximum degree in T_i .

Suppose that $|V_i| \leq 2$ for some i , $1 \leq i \leq n/3$. Since T is a tree of order at least 3, we may assume that v_i is adjacent to a vertex $u \in V_j$ for some $j \neq i$. Note that uv_i is the only edge joining V_i and V_j . If $u = v_j$, then $T_i \cup T_j \cup \{uv_i\}$ is a spider and $(P - \{V_i, V_j\}) \cup \{V_i \cup V_j\}$ is a spider partition with fewer than $n/3$ sets, contradicting the fact that S is a minimum spider partition. If u is an endvertex in T_j , then again $T_i \cup T_j \cup \{uv_i\}$ is a spider, and we have the same contradiction. Hence, u must be a vertex of degree 2 in T_j . Thus v_j has degree at least 3 in T_j . (For otherwise we could choose u as the vertex of maximum degree in T_j , which produces a contradiction.) We may assume that the tree T_j is rooted at the vertex v_j and that u_j is the parent of u in T_j (possibly, $u_j = v_j$). Let T'_j be the maximal subtree of T_j rooted at u (i.e., T'_j consists of u and its descendants in T_j). Then, $T_i \cup T'_j \cup \{uv_i\}$ is a spider of order at least 4, while $T_j - V(T'_j)$ is a spider of order at least 3. Let $V'_i = V_i \cup V(T'_j)$, and let $V'_j = V_j - V(T'_j)$. Then, $(P - \{V_i, V_j\}) \cup \{V'_i, V'_j\}$ is a spider partition with cardinality $n/3$ having fewer sets of cardinality at most 2 than P , contradicting our choice of P . Hence, $|V_i| \geq 3$ for each i , $1 \leq i \leq n/3$. Since $\gamma_P(T) = n/3$, each set V_i therefore has cardinality exactly 3; i.e., $T_i = P_3$ for $1 \leq i \leq n/3$. If, in T , an endvertex of T_i has a neighbor in a set V_j , $j \neq i$, then $V_i \cup V_j$ induces a spider and again we have a smaller spider partition, a contradiction. Hence, the $n/3 - 1$ edges connecting the spider subgraphs to form T are incident to the vertices in S . It follows that $T = T' \circ \overline{K}_2$, where T' is the tree induced by the vertices in S . \square

Next we present a linear algorithm for finding a minimum PDS in a nontrivial tree T .

ALGORITHM 1.

Input: A tree G on $n \geq 2$ vertices rooted at a vertex of maximum degree with the vertices labeled v_1, v_2, \dots, v_n so that $\ell(v_i) \leq \ell(v_j)$ for $i > j$. [Note: the root of G is labeled v_n .]

Output: A minimum PDS S of G and a partition of $V(G)$ into $|S|$ subsets $\{V_x \mid x \in S\}$ so that each subset induces a spider.

Begin

1. If G is a spider, then $S \leftarrow \{v_n\}$ and $V_{v_n} \leftarrow V(G)$, and output S and $\{V_x \mid x \in S\}$; otherwise, continue.
2. $Type(v_i) \leftarrow \text{TRUE}$ and $V_{v_i} \leftarrow \emptyset$ for all $i = 1, 2, \dots, n$.
3. $i \leftarrow 1$, $T \leftarrow G$, $\mathcal{I} \leftarrow \{1, 2, \dots, n\}$, and $S \leftarrow \emptyset$.
4. $v \leftarrow v_i$.
5. If $\deg_T v \leq 2$, then
 - 5.1. if there exists a child u of v (in G) such that $Type(u) = \text{TRUE}$ and $u \in V_x$ for some $x \in S$, then
 - 5.1.1. if v is a leaf (in T), then
 - 5.1.1.1. $V_x \leftarrow V_x \cup \{v\}$,
 - 5.1.1.2. $T \leftarrow T - v$, $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$ and go to step 13;
 - 5.1.2. if $\deg_T v = 2$, then
 - 5.1.2.1. $V_x \leftarrow V_x \cup V(T_v)$,
 - 5.1.2.2. $w \leftarrow v$ and go to step 12;
 - 5.2. otherwise (if no such child exists), $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\}$ and go to step 13; otherwise (if $\deg_T v \geq 3$), then continue.

6. $S \leftarrow S \cup \{v\}$.
7. $w \leftarrow v_m$, where $m \leftarrow \min\{k \mid v_k \text{ is an ancestor of } v \text{ of degree at least 3 in } T \text{ or } k = n\}$.
8. $u \leftarrow \{\text{child of } w \text{ on the } w\text{-}v \text{ path}\}$.
9. If $w = v_n$, then
 - 9.1. if the component of $T - uw$ containing w is the trivial path w or a path with w as a leaf, then $V_v \leftarrow V(T_w)$, and output S and $\{V_x \mid x \in S\}$;
 - 9.2. otherwise, $V_v \leftarrow V(T_u)$ and go to step 11.
10. If $w \neq v_n$, then
 - 10.1. $z \leftarrow \text{parent}(w)$;
 - 10.2. if $\deg_T w \geq 4$ or if $\deg_T w = 3$ and the component of $T - \{uw, wz\}$ containing w is not a path, then $V_v \leftarrow V(T_u)$ and go to step 11;
 - 10.3. if $\deg_T w = 3$ and the component of $T - \{uw, wz\}$ containing w is a path, then $V_v \leftarrow V(T_w)$ and go to step 12.
11. $T \leftarrow T - V(T_u)$, $\mathcal{I} \leftarrow \mathcal{I} \setminus \{k \mid v_k \in V(T_u)\}$, and go to step 13.
12. $T \leftarrow T - V(T_w)$, $\mathcal{I} \leftarrow \mathcal{I} \setminus \{k \mid v_k \in V(T_w)\}$, $\text{Type}(w) \leftarrow \text{FALSE}$. Go to step 13.
13. $i \leftarrow \min\{k \mid k \in \mathcal{I}\}$.
14. If $i < n$, then return to step 4.
15. If $i = n$, then
 - 15.1. if (a) T is the trivial path v_n or T is a path with v_n as a leaf, and (b) there exists a child u of v (in G) such that $\text{Type}(u) = \text{TRUE}$ and $u \in V_x$ for some $x \in S$, then
 - 15.1.1. $V_x \leftarrow V_x \cup V(T_{v_n})$,
 - 15.1.2. output S and $\{V_x \mid x \in S\}$.
 - 15.2. Otherwise (if (a) or (b) in 15.1 does not hold), then
 - 15.2.1. $S \leftarrow S \cup \{v_n\}$,
 - 15.2.2. $V_{v_n} \leftarrow V(T_{v_n})$,
 - 15.2.3. output S and $\{V_x \mid x \in S\}$.

End

We now verify the validity of Algorithm 1.

THEOREM 15. *Algorithm 1 produces a $\gamma_P(G)$ -set in a nontrivial tree G .*

Proof. Let G be a nontrivial tree of order n , and let S be the set produced by Algorithm 1. It follows from the way in which the set S is constructed that S is a PDS of G , and so $\gamma_P(G) \leq |S|$. We show that $\gamma_P(G) = |S|$. If G is a spider, then it follows from Theorem 7 and from our choice of the root v_n that $S = \{v_n\}$ is a $\gamma_P(G)$ -set of G . Hence, we may assume that $\gamma_P(G) \geq 2$. Let $S = \{v_{i_1}, v_{i_2}, \dots, v_{i_m}\}$, where $i_1 < i_2 < \dots < i_m$. Suppose that $\gamma_P(G) < |S|$. Among all $\gamma_P(G)$ -sets, let S^* be chosen so that the first integer j , $1 \leq j \leq m$, such that $v_{i_j} \notin S^*$ is as large as possible. If $|S^* \cap V_{i_t}| \geq 2$ for some t , $1 \leq t < j$, then any vertex of $S^* \cap V_{i_t}$ different from $\{v_{i_t}\}$ can be replaced with an ancestor of v_{i_t} that does not belong to V_{i_t} . Hence, we may assume that for each t with $1 \leq t < j$, $S^* \cap V_{i_t} = \{v_{i_t}\}$. If S^* contains a descendant u of v_{i_j} that belongs to V_{i_j} , then we can replace u with v_{i_j} to produce a new $\gamma_P(G)$ -set that contains all the vertices in $\{v_{i_1}, \dots, v_{i_j}\}$, contrary to our choice of S^* . Hence, S^* contains no descendant of v_{i_j} that belongs to V_{i_j} .

We now consider the subtree T which has been constructed at the stage in the algorithm when v_{i_j} is considered as the active vertex v in step 5. Since v_{i_j} was added to S in step 6 of the algorithm, we know that $\deg_T v_{i_j} \geq 3$, and so v_{i_j} has at least two children in T . Let y be a descendant of v_{i_j} in T . Since y has not already been

deleted from T , it follows from step 5 of the algorithm that every child u of y (in G) that belongs to $V(G) - V(T)$ satisfies $Type(u) = FALSE$ and therefore has degree at least 3 (in G) and does not belong to S . Since S^* contains no descendant of v_{i_j} that belongs to V_{i_j} , it is evident that y cannot be observed by S^* , a contradiction. We deduce, therefore, that we must have $\gamma_P(G) = |S|$. \square

In closing, we note that we have yet to establish the power domination number of graphs of bounded tree width.

REFERENCES

- [1] T. L. BALDWIN, L. MILI, M. B. BOISEN, JR., AND R. ADAPA, *Power system observability with minimal phasor measurement placement*, IEEE Trans. Power Syst., 8 (1993), pp. 707–715.
- [2] M. B. BOISEN, JR., T. L. BALDWIN, AND L. MILI, *Simulated Annealing and Graph Theory Applied to Electrical Power Networks*, manuscript, 2000.
- [3] D. J. BRUENI, *Minimal PMU Placement for Graph Observability: A Decomposition Approach*, Masters thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1993.
- [4] T. W. HAYNES, S. T. HEDETNIEMI, AND P. J. SLATER, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.
- [5] T. W. HAYNES, S. T. HEDETNIEMI, AND P. J. SLATER, EDs., *Domination in Graphs: Advanced Topics*, Marcel Dekker, New York, 1998.
- [6] L. MILI, T. BALDWIN, AND A. PHADKE, *Phasor measurement placement for voltage and stability monitoring and control*, in Proceedings of the EPRI-NSF Workshop on Application of Advanced Mathematics to Power Systems, San Francisco, CA, 1991.

ROBUST MATCHINGS*

REFAEL HASSIN[†] AND SHLOMI RUBINSTEIN[†]

Abstract. We consider complete graphs with nonnegative edge weights. A p -matching is a set of p disjoint edges. We prove the existence of a maximal (with respect to inclusion) matching M that contains for any $p \leq |M|$ p edges whose total weight is at least $\frac{1}{\sqrt{2}}$ of the maximum weight of a p -matching. We use this property to approximate the metric maximum clustering problem with given cluster sizes.

Key words. robust matching, maximum clustering

AMS subject classifications. 05C70, 05C85

PII. S0895480198332156

1. Introduction. Let $G = (V, E)$ be a complete graph with vertex set V such that $|V| = n$, edge set E , and edge weights $w(u, v) \geq 0$, $(u, v) \in E$. A p -matching is a set of p disjoint edges in a graph. A p -matching with $p = \lfloor \frac{n}{2} \rfloor$ is called perfect. A perfect matching M that for each $p = 1, \dots, |M|$ contains p edges whose total weight is at least α times the maximum weight of a p -matching is said to be α -robust. We prove that G contains a $\frac{1}{\sqrt{2}}$ -robust matching. On the other hand, there are graphs that do not contain an α -robust matching for any $\alpha > \frac{1}{\sqrt{2}}$.

In section 2, we generalize the robustness concept to independence systems. We prove that a greedy algorithm can be applied in some cases to form robust solutions. Our main theorem on robust matchings is proved in section 3, and we use it and a theorem from section 2 to approximate within a factor $\frac{1}{\sqrt{2}}$ the following problem: Given constants $c_1 \geq c_2 \geq \dots \geq c_p$, find a p -matching M that maximizes $\sum_{i=1}^p c_i w_i$, where $w_1 \geq w_2 \geq \dots \geq w_p$ are the edge weights in M .

In section 4, we use these results to approximate the NP-hard METRIC MAXIMUM CLUSTERING PROBLEM WITH GIVEN CLUSTER SIZES. The input for the problem is a complete graph with edge weights that satisfy the triangle inequality, and a set of cluster sizes. The goal is to find a partition of the vertex set, with part (or “cluster”) sizes as required, that maximizes the total edge weight within the same cluster.

For $V' \subseteq V$ we denote by $E(V')$ the edge set of the subgraph induced by V' . For $E' \subseteq E$ we denote by $W(E')$ the total weight of edges in E' .

2. Robust independent sets. An *independence system* is a pair (E, \mathcal{F}) consisting of a ground set E and a collection of *independent sets*, or, equivalently, *feasible solutions*, such that $F' \subset F \in \mathcal{F}$ implies $F' \in \mathcal{F}$. Let $w_e \geq 0$, $e \in E$, be weights attached to the elements of E . The problem of computing an independent set of maximum weight generalizes many interesting combinatorial optimization problems. Korte and Hausmann [1] analyzed the performance of the greedy algorithm for the above problem. The algorithm sorts the elements by weight and inserts them into the solution, starting with the heaviest one and excluding an element if its addition would generate a set not in \mathcal{F} . They proved the following theorem.

*Received by the editors January 5, 1998; accepted for publication (in revised form) July 1, 2002; published electronically September 10, 2002.

<http://www.siam.org/journals/sidma/15-4/33215.html>

[†]Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel (hassin@post.tau.ac.il, shlomiru@post.tau.ac.il).

THEOREM 2.1. *For any $E' \subseteq E$ define $l(E')$ and $u(E')$ to be the smallest and largest cardinality, respectively, of a maximal (with respect to inclusion) independent set contained in E' . Let $r(E, \mathcal{F}) = \min_{E' \subseteq E} \frac{l(E')}{u(E')}$. Then the greedy solution is an $r(E, \mathcal{F})$ -approximation; that is, the value of the greedy solution is at least $r(E, \mathcal{F})$ times the optimal value.*

Now consider a game of the following type: You choose a maximal independent set in E . An adversary then selects a bound p on the allowed number of elements. Last, you output the p heaviest elements of your solution (or the solution itself if p is greater or equal to its cardinality). By the definition of an independence system, the output is independent. Your payoff is the ratio between the weight of your output and the maximum weight of an independent set whose cardinality is at most p . A solution is α -robust if it guarantees a payoff of at least α .

THEOREM 2.2. *The greedy solution is $r(E, \mathcal{F})$ -robust.*

Proof. Define (E, \mathcal{F}_p) to be the independence system in which $F \in \mathcal{F}_p$ if and only if $F \in \mathcal{F}$ and $|F| \leq p$. Let l_p and u_p denote the l and u values in the new system, respectively. Then for every $E' \subseteq E$

$$\frac{l_p(E')}{u_p(E')} = \frac{\min(l(E'), p)}{\min(u(E'), p)} \geq \frac{l(E')}{u(E')}$$

so that $r(E, \mathcal{F}_p) \geq r(E, \mathcal{F})$ and the claim follows from Theorem 2.1. \square

The edges and matchings in a graph constitute an independence system for which $r = \frac{1}{2}$ [1]. It follows that the greedy solution is $\frac{1}{2}$ -robust. We will reach this conclusion later in a different way but not before we obtain stronger results in the next section.

2.1. Weighted robustness. Let $c_1 \geq c_2 \geq \dots \geq c_n \geq 0$ be given constants. For an independent set $F = \{e_1, \dots, e_m\}$ with weights w_1, w_2, \dots, w_m , define $C(F) = \sum_{j=1}^m c_j w_j$. Since we are interested in obtaining large values of $C(F)$, we will assume that the elements are numbered so that $w_1 \geq w_2 \geq \dots \geq w_m$. Thus, $C(F)$ is well defined for any set F without explicitly specifying an order on its elements. We will also denote $F_p = \{e_1, \dots, e_p\}$, $p = 1, \dots, m$, and $F_p = F$ for $p > m$.

PROBLEM 2.3. *Compute an independent set $F \in \mathcal{F}$ of cardinality $|F| \leq p$ that maximizes $C(F_p)$.*

The following theorem was proved by Gerhard Woeginger.

THEOREM 2.4. *Problem 2.3 is NP-hard even when \mathcal{F} is the set of matchings in a graph with edge set E (so that $F \subseteq E$ is in \mathcal{F} if it consists of vertex-disjoint edges).*

Proof. The reduction is from the following NP-complete variant of 3-PARTITION.

Input: A positive integer t . $2n$ positive integers a_1, a_2, \dots, a_{2n} and n positive integers b_1, b_2, \dots, b_n . These integers fulfill the equation $\sum_{i=1}^{2n} a_i + \sum_{i=1}^n b_i = nt$. Moreover, $a_i \leq t$ and $b_i \leq t$ holds for all i . These $3n$ integers are encoded in *unary*.

Question: Does there exist a permutation π of $\{1, \dots, 2n\}$ such that for all $i = 1, \dots, n$ we have $a_{\pi(2i-1)} + a_{\pi(2i)} + b_i = t$?

Consider an instance I of 3-PARTITION. For $i = 1, \dots, n$ we define the cost coefficient $c_i = n^{b_i}$. Moreover, we construct from I an edge-weighted complete graph G on $2n$ vertices v_1, \dots, v_{2n} . The weight $w(v_i, v_j)$ of the edge between vertices v_i and v_j equals

$$w(v_i, v_j) = n^{2t} - n^{a_i + a_j} \geq 0.$$

All these weights and costs are encoded in binary. Then the length of the encoding of every such value is bounded by $2t \log n$, which is polynomial in the size of the instance I (which is encoded in unary). We claim that G has a matching F of n independent edges with

$$C(F) \geq n^{2t} \sum_{i=1}^n n^{b_i} - n^{t+1} =: T^*$$

if and only if the instance I of 3-PARTITION has answer YES.

(If) Consider a solution π of 3-PARTITION. For $i = 1, \dots, n$ the matching F matches vertex $v_{\pi(2i-1)}$ with vertex $v_{\pi(2i)}$ and assigns the cost coefficient c_i to this edge. Hence this edge contributes $n^{2t}n^{b_i} - n^t$ to the objective value, and the claim follows.

(Only if) Consider a matching F with the desired objective value. For $i = 1, \dots, n$ let $(v_{\pi(2i-1)}, v_{\pi(2i)})$ denote the edge that is assigned to the cost coefficient $c_i = n^{b_i}$. We claim that π constitutes a solution to the 3-PARTITION instance. Otherwise, there exists an index i with $a_{\pi(2i-1)} + a_{\pi(2i)} + b_i \geq t + 1$. The contribution of this coefficient is then $\leq n^{b_i}n^{2t} - n^{t+1}$. The contribution of every other coefficient c_j is at most $n^{b_j}(n^{2t} - n)$. We can never reach an objective value T^* . \square

THEOREM 2.5. *Let F and F' be independent sets. If F' is α -robust, then $C(F'_p) \geq \alpha C(F_p)$ for every $p = 1, 2, \dots, n$ and any constants $c_1 \geq c_2 \geq \dots \geq c_n \geq 0$.*

Proof. Let $w_1 \geq w_2 \geq \dots \geq w_p$ and $w'_1 \geq w'_2 \geq \dots \geq w'_p$ be the weights of the elements of F_p and F'_p , respectively. (If $|F| < p$, then define $w_j = 0$ for $j > |F|$.) Then

$$\begin{aligned} C(F'_p) &= \sum_{i=1}^p c_i w'_i \\ &= \sum_{j=1}^{p-1} (c_j - c_{j+1}) \sum_{i=1}^j w'_i + c_p \sum_{i=1}^p w'_i \\ &= \sum_{j=1}^{p-1} (c_j - c_{j+1}) W(F'_j) + c_p W(F'_p) \\ &\geq \sum_{j=1}^{p-1} (c_j - c_{j+1}) \alpha W(F_j) + c_p \alpha W(F_p) \\ &= \alpha \sum_{i=1}^p c_i w_i = \alpha C(F_p). \quad \square \end{aligned}$$

3. Robust matchings. A *matching* is a set of vertex-disjoint edges. The weight of a matching is the total weight of its edges. A *maximum matching* is a matching with maximum weight. A *p-matching* is a matching with p edges. We denote $m = \lfloor \frac{n}{2} \rfloor$, the maximum number of edges in a matching. An m -matching is said to be *perfect*. Note that this extends the common use of this concept to the case where the graph has an odd number of vertices.

An easy way to produce a maximum p -matching is as follows: Extend G by adding to it $n - 2p$ vertices. Each new vertex is connected to each original vertex by an edge with a “large” weight, say, twice the largest weight in G . Now compute a maximum

perfect matching (with $n - p$ edges) in the extended graph. It will be composed of $n - 2p$ heavy new edges and a maximum p -matching in G .

For a perfect matching M we define M_p to be the set of its p heaviest edges, $p = 1, \dots, m$. We denote by $M^{(p)}$ a maximum p -matching. A matching is α -robust if

$$W(M_p) \geq \alpha W(M^{(p)}) \quad p = 1, \dots, m.$$

In this section, we show that for every graph there exists a $\frac{1}{\sqrt{2}}$ -robust matching and that it can be constructed by a single application of a maximum matching algorithm. The following example shows that the value of $\frac{1}{\sqrt{2}}$ cannot be increased.

Consider a 4-vertex graph with weights $w(1, 2) = w(3, 4) = 1$, $w(2, 3) = \sqrt{2}$, and all other edges have zero weight. For this graph $W(M_1) = \sqrt{2}$ and $W(M_2) = 2$. The graph has three perfect matchings and none is α -robust for $\alpha > \frac{1}{\sqrt{2}}$: The matchings $\{(1, 2), (3, 4)\}$ and $\{(2, 3), (1, 4)\}$ are $\frac{1}{\sqrt{2}}$ -robust, and $\{(1, 3), (2, 4)\}$ is 0-robust.

THEOREM 3.1. *Let S be a maximum perfect matching with respect to the squared weights $w^2(e)$, $e \in E$. S is $\frac{1}{\sqrt{2}}$ -robust.*

The rest of this section is devoted to proving Theorem 3.1. We will prove it by treating the squared edge weights as variables whose sizes are to be determined in order to form a contradiction to the theorem. We will prove that to achieve such a contradiction we may make several assumptions on these variables. Finally, these assumptions will lead to the conclusion that the claim is true.

Consider the set $S \cup M^{(p)}$. It consists of a collection of disjoint paths and cycles. A path may consist of a single edge or it alternates between S and $M^{(p)}$. Since S is perfect, the end edges of the path are from S except possibly one end of one path in the case of odd n (since in this case there is exactly one vertex that is not incident to an edge of S). A cycle alternates between S and $M^{(p)}$. We will construct from the edges of S a p -matching whose weight is at least $\frac{W(M^{(p)})}{\sqrt{2}}$. Since the weight of this matching is at most the weight of the p heaviest edges in S , this construction will prove the theorem.

We choose a p -matching from S as follows: Every edge in $S \cap M^{(p)}$ is chosen. All of the edges of S contained in a cycle of $S \cup M^{(p)}$ are chosen. From every nontrivial path (containing more than a single edge) of $S \cup M^{(p)}$ we choose all the edges that belong to S except for the lightest one. There is one exception to the last rule: If there is a path with only one end edge from S (this happens when n is odd), then we choose all of the S -edges of this path. The total number of edges selected is equal to $|M^{(p)}| = p$. It is sufficient to prove that the claimed bound on the ratio of the edge weights in S and in $M^{(p)}$ holds for every such path and cycle.

Consider a nontrivial path P with squared weights $x_1, y_1, x_2, y_2, \dots, y_{r-1}, x_r$, where the x values correspond to the edges of S and the y values correspond to the edges of $M^{(p)}$ in the order they appear on P .

We denote $x_{[i,j]} = \sum_{l=i}^j x_l$, and similarly $y_{[i,j]} = \sum_{l=i}^j y_l$. We are interested in subpaths $P_{i,j}$ of P consisting of the edges whose weights are $x_i, y_i, \dots, y_{j-1}, x_j$. Note that $P = P_{1,r}$. Since S is maximum with respect to the squared weights,

$$(3.1) \quad x_{[i,j]} \geq y_{[i,j-1]} \quad 1 \leq i < j \leq r.$$

Let $x_{\min} = \min\{x_i \mid i = 1, \dots, r\}$. Our goal is to prove that the ratio of the total weight of the $r - 1$ heaviest edges in $P \cap S$ to the weight of $P \cap M_k$ is at least $\frac{1}{\sqrt{2}}$;

that is,

$$Z = \frac{\sum_{i=1}^r \sqrt{x_i} - \sqrt{x_{\min}}}{\sum_{i=1}^{r-1} \sqrt{y_i}} \geq \frac{1}{\sqrt{2}}$$

for all x, y that satisfy (3.1).

We will prove that $Z \geq \frac{1}{\sqrt{2}}$ for every nontrivial path by induction on r . Note that the proof and induction hypothesis apply to any nontrivial path P in $S \cup M^{(p)}$, not just to maximal (with respect to inclusion) paths. A subpath is subject to additional constraints arising from longer subpaths that contain it, but these constraints may increase only the lower bound on Z for the subpath in question. We first prove this property for $r = 2$ and $r = 3$.

LEMMA 3.2. $Z \geq \frac{1}{\sqrt{2}}$ when $r = 2$.

Proof. Without loss of generality we can assume that $x_1 \geq x_2$; thus $Z = \frac{\sqrt{x_1}}{\sqrt{y_1}}$, and we look for its minimum subject to $x_1 + x_2 \geq y_1$ and $x_1 \geq x_2$. This minimum is obtained when $x_1 = x_2 = \frac{y_1}{2}$ and its value is $\frac{1}{\sqrt{2}}$. \square

LEMMA 3.3. $Z \geq \frac{1}{\sqrt{2}}$ when $r = 3$.

Proof. In this case x_2 appears in every constraint of (3.1), and thus Z can be minimized with $x_2 \geq x_1, x_3$. Without loss of generality we assume that $x_3 = x_{\min}$ so that $Z = \frac{\sqrt{x_1} + \sqrt{x_2}}{\sqrt{y_1} + \sqrt{y_2}}$. We minimize Z subject to (3.1) and $x_2 \geq x_1 \geq x_3$. If $x_1 > x_{\min}$, then, by concavity of the square root function, Z can be reduced by decreasing x_1 and increasing x_2 . Thus we assume $x_1 = x_3 = x_{\min}$. Again by concavity, Z can be decreased by increasing x_2 and simultaneously decreasing x_1 and x_3 . This change is feasible if $x_1 + x_2 + x_3 > y_1 + y_2$. Thus we assume equality in this constraint, that is, $2x_1 + x_2 = y_1 + y_2$. We now show that the claimed bound holds even for the relaxed problem of minimizing Z subject to only $2x_1 + x_2 = y_1 + y_2$ and $x_2 \geq x_1 \geq 0$. Suppose first that y_1 and y_2 are fixed and we minimize over x_1 and x_2 . The feasible set of solutions is a convex polyhedron and the objective function, $\sqrt{x_1} + \sqrt{x_2}$, is concave. Thus the minimum value is attained at an extreme point of the feasible set. There are two such points. In one, $x_1 = x_2 = \frac{y_1 + y_2}{3}$ and $Z = \frac{2\sqrt{\frac{y_1 + y_2}{3}}}{\sqrt{y_1} + \sqrt{y_2}}$. In the other one, $x_1 = 0$ and $x_2 = y_1 + y_2$, giving $Z = \frac{\sqrt{y_1 + y_2}}{\sqrt{y_1} + \sqrt{y_2}}$, which is clearly smaller than the former case and attains its minimum over y_1 and y_2 when $y_1 = y_2$ and $Z = \frac{1}{\sqrt{2}}$. \square

We now proceed to proving the general step of the induction for $r > 3$. Thus, we assume that the claim holds for smaller r values.

LEMMA 3.4. We can assume that $x_j > x_{\min}$ $j = 2, \dots, r - 1$.

Proof. Suppose that $x_j = x_{\min}$ for some $j \in \{2, \dots, r - 1\}$. Then

$$\begin{aligned} Z &= \frac{(\sum_{i=1}^j \sqrt{x_i} - \sqrt{x_{\min}}) + (\sum_{i=j}^r \sqrt{x_i} - \sqrt{x_{\min}})}{\sum_{i=1}^{j-1} \sqrt{y_i} + \sum_{i=j}^{r-1} \sqrt{y_i}} \\ &\geq \min \left\{ \frac{\sum_{i=1}^j \sqrt{x_i} - \sqrt{x_{\min}}}{\sum_{i=1}^{j-1} \sqrt{y_i}}, \frac{\sum_{i=j}^r \sqrt{x_i} - \sqrt{x_{\min}}}{\sum_{i=j}^{r-1} \sqrt{y_i}} \right\}. \end{aligned}$$

Since $x_j = \min\{x_i \mid i = 1, \dots, j\} = \min\{x_i \mid i = j, \dots, r\}$, it follows from the induction hypothesis that $Z \geq \frac{1}{\sqrt{2}}$. \square

We call a subpath $P_{i,j}$ for which $x_{[i,j]} = y_{[i,j-1]}$ *tight*.

LEMMA 3.5. (i) Let $i \leq k \leq j \leq l$ such that $i < j$ and $k < l$. If $k < j$ and both $P_{i,j}$ and $P_{k,l}$ are tight, then so is $P_{k,j}$. (ii) Let $i < j < l$. If P_{ij} is tight, then $P_{j,l}$ is not.

Proof. (i) By assumption, $x_{[i,j]} = y_{[i,j-1]}$ and $x_{[k,l]} = y_{[k,l-1]}$. Summing these equations we get

$$x_{[i,l]} + x_{[k,j]} = x_{[i,j]} + x_{[k,l]} = y_{[i,j-1]} + y_{[k,l-1]} = y_{[i,l-1]} + y_{[k,j-1]}.$$

Since $x_{[i,l]} \geq y_{[i,l-1]}$ and $x_{[k,j]} \geq y_{[k,j-1]}$ it follows that both of the latter relations satisfy equality and the respective subpaths are tight.

(ii) From the same equation with $j = k$ it follows that $x_j = 0$ and $1 < j < r$, in contrast to Lemma 3.4. \square

Suppose that $r \geq 3$. Let $1 < j < r$. We can assume that there exists a tight interval containing the edge e_j whose weight is x_j ; otherwise, we reduce x_j until some subinterval containing e_j becomes tight, and this change reduces Z . Consider the intersection of all tight intervals containing $e_j \in S$. It follows from Lemma 3.5 that the intersection is a nontrivial tight subpath. Again by this lemma, the x values in this subpath share the same set of tight subpaths, and therefore we can assume that the sum of their squared roots is minimized subject to a single constraint on their sum. By concavity of the square root function, this objective is attained by setting all of these values to 0 except for a single one, say $x_k > 0$. From Lemma 3.4 and since $x_{\min} \geq 0$, it follows that $k \leq 4$. For $r \leq 3$, the claim has already been proved in Lemmas 3.2 and 3.3. Suppose that $r = 4$; then it must be that P_{12} and P_{34} are tight, and thus $x_1 = x_4 = y_2 = 0$, $y_1 = x_2$, and $y_3 = x_3$. In this case $Z = 1$, and this completes the proof for paths with two ends from S .

For a path with only one end edge from S we can assume that a fictitious S -edge of zero weight is added at that end. The set of constraints (3.1) then extends in a natural way, and the same proof holds.

Now suppose that there is a cycle C that contradicts the claim. We will show how to construct an instance consisting of a path that contradicts the claim. Since we have already proved that this is impossible, it will follow that such a cycle cannot exist. Specifically, let the cycle's edges have weights $x_1, y_1, \dots, x_r, y_r$, in this order, with the x -weights corresponding to edges of S . Form a path by concatenating many repetitions of this sequence of weights. Last, add an x -edge at the end where it is missing, with a sufficiently large weight, such as $W(C \cap M^{(p)})$, so that (3.1) is satisfied. The path obtained this way will have (asymptotically, as the number of pasted copies increases) the same Z -value as C . This concludes the proof of Theorem 3.1.

3.1. More robustness results. Most of the proof of Theorem 3.1 is valid for any concave function, not just the square root function. Using this observation, the theorem can be generalized as follows. Let S_b be a maximum perfect matching with respect to the weights $w^b(e)$, $e \in E$, where $b \geq 1$ is a constant. Let $\beta = \frac{1}{b}$. The following lemmas and theorem follow easily by adapting the proofs of the respective results obtained in the previous subsection for $\beta = \frac{1}{2}$.

LEMMA 3.6. $Z \geq \frac{1}{2^\beta}$ when $r = 2$.

LEMMA 3.7. $Z \geq \frac{1}{2^{1-\beta}}$ when $r = 3$.

THEOREM 3.8. S_b is $\min\{\frac{1}{2^\beta}, \frac{1}{2^{1-\beta}}\}$ -robust.

Maximum robustness is obtained when $\beta = \frac{1}{2}$, the case to which Theorem 3.1 applies. We note two interesting extreme cases. When $b = \beta = 1$, S_b is just a matching of maximum weight. When $b \rightarrow \infty$ and thus $\beta \rightarrow 0$, we get a greedy matching. Such

a matching is obtained by sorting the edge weights in nonincreasing order, and then scanning the list and adding an edge to the matching if it is disjoint to the previously selected edges. In both cases the resulting bound is $\frac{1}{2}$, giving the following corollary.

COROLLARY 3.9. *Maximum and greedy matchings are $\frac{1}{2}$ -robust.*

4. Clustering. In the METRIC MAXIMUM CLUSTERING PROBLEM WITH GIVEN CLUSTER SIZES, the goal is to partition the vertex set V into sets (“clusters”) of given sizes so that the total weight of edges inside the clusters is maximized. Specifically, the input for the problem consists of a complete graph with edge weights satisfying the triangle inequality and cluster sizes $c_1 \geq c_2 \geq \dots \geq c_p \geq 1$ such that $c_1 + \dots + c_p = n$. We want to partition V into clusters of these sizes maximizing their total weight.

Let $d_j = \lfloor \frac{c_j}{2} \rfloor$, $D_j = d_1 + \dots + d_j$ $j = 1, \dots, p$, and $D_0 = 0$. We propose the following algorithm.

ALGORITHM 4.1.

1. Compute a maximum matching S with respect to the squared weights. Let $S = \{(u_j, v_j) \mid j = 1, \dots, m\}$, where $w(u_j, v_j) \geq w(u_{j+1}, v_{j+1})$ $j = 1, \dots, m-1$.
2. Set $V_i = \{u_j, v_j \mid j = D_{i-1} + 1, \dots, D_i\}$ $i = 1, \dots, p$.
3. For each i such that c_i is odd, add to V_i an arbitrary yet unassigned vertex.

THEOREM 4.2. *Let opt and apx denote the solution values of the optimal and approximate solutions, respectively. Then*

$$apx \geq \frac{1}{2\sqrt{2}}opt.$$

Proof. Consider an optimal partition O_1, \dots, O_p . Let M_i be a maximum matching in the subgraph induced by O_i , $i = 1, \dots, p$. Denote the edge weights in M_i by $w_1^i \geq \dots \geq w_{d_i}^i$.

Let $b_i = c_i - 1$ if c_i is even and $b_i = c_i$ if c_i is odd. The edges $E(O_i)$ can be covered by a set of $b_i \leq c_i$ disjoint matchings. Since M_i is a maximum matching in G_i it follows that $b_i W(M_i) \geq W(E(O_i))$ and therefore

$$opt \leq \sum_{i=1}^p c_i W(M_i).$$

Let V_1, \dots, V_p be the partition produced by Algorithm 4.1. Let $S_i = S \cap E(V_i)$. Consider a cluster V_i with vertices $u, v, q \in V_i$ such that $(u, v) \in S_i$. By the triangle inequality, $w(u, q) + w(v, q) \geq w(u, v)$.

Suppose that c_i is even. Sum this inequality over all $q \neq u, v \in V_i$; then sum again over $(u, v) \in S_i$. Note that every edge in $E(V_i) \setminus S_i$ is summed twice. Thus, every edge $(u, v) \in S_i$ contributes to the total weight of $E(V_i)$ in addition to its own weight also at least $\frac{1}{2}(c_i - 2)$ times its weight through the edges incident to it. Thus, $W(E(V_i)) \geq \frac{1}{2}c_i W(S_i)$.

Suppose now that c_i is odd. In this case V_i contains a vertex, say v_i , that was added to V_i in step 3 of the algorithm. In the summation, the weight of edges incident to v_i is used just once. Thus, each edge $(u, v) \in S_i$ contributes its weight $\frac{1}{2}(c_i - 3)$ times when summed over $V_i \setminus \{u, v, v_i\}$, once more through $w(u, v_i) + w(v, v_i)$, and once when it contributes its own weight. Thus, also in this case, $W(E(V_i)) \geq \frac{1}{2}c_i W(S_i)$.

By Theorem 2.5 and the assumption $c_1 \geq \dots \geq c_p$,

$$\begin{aligned} \text{apx} &\geq \frac{1}{2} \sum_{i=1}^p c_i W(S_i) \\ &\geq \frac{1}{2\sqrt{2}} \sum_{i=1}^p c_i W(M_i) \\ &\geq \frac{1}{2\sqrt{2}} \text{opt}. \quad \square \end{aligned}$$

Acknowledgment. We thank Gerhard Woeginger for permitting us to include his proof of Theorem 2.4.

REFERENCES

- [1] B. KORTE AND D. HAUSMANN, *An analysis of the greedy heuristic for independence systems*, Ann. Discrete Math., 2 (1978), pp. 65–74.

CHOOSABILITY AND EDGE CHOOSABILITY OF PLANAR GRAPHS WITHOUT INTERSECTING TRIANGLES*

WEL-FAN WANG[†] AND KO-WEI LIH[‡]

Abstract. Let G be a planar graph without two triangles sharing a common vertex. We prove that (1) G is 4-choosable and (2) G is edge- $(\Delta(G)+1)$ -choosable when its maximum degree $\Delta(G) \neq 5$.

Key words. choosability, edge choosability, planar graph

AMS subject classification. 05C15

PII. S0895480100376253

1. Introduction. We consider only simple graphs in this paper unless otherwise stated. A *plane* graph is a particular drawing of a planar graph in the Euclidean plane. For a plane graph G , we denote its vertex set, edge set, face set, order, and minimum degree by $V(G)$, $E(G)$, $F(G)$, $|G|$, and $\delta(G)$, respectively. Let $d_G(x)$ denote the degree of a vertex or a face x of G . The degree of a face is the total number of edge steps of the closed walk(s) in G bounding the face. When the boundary of a face f is a cycle, we usually write $f = [u_1u_2 \cdots u_n]$, where u_1, u_2, \dots, u_n are the boundary vertices in cyclic order. A vertex (or face) of degree k is called a k -vertex (or k -face). A *triangle* is synonymous with a 3-cycle. We say that two cycles or faces of a plane graph are *adjacent* if they share at least one common (boundary) edge. Two cycles or faces are *intersecting* if they share at least one common (boundary) vertex. A vertex v and a face f are said to be *incident* if v belongs to the boundary of f . Let $F(v)$ denote the set of all faces that are incident to the vertex v .

A *coloring* of a graph G is a mapping ϕ from $V(G)$ to the set of colors $\{1, 2, \dots, k\}$ for some positive integer k . A coloring is called *proper* if $\phi(x) \neq \phi(y)$ for every edge xy of G . The *chromatic number* $\chi(G)$ is the smallest integer k such that G has a proper coloring into the set $\{1, 2, \dots, k\}$. We say that L is an *assignment* for the graph G if it assigns a list $L(v)$ of possible colors to each vertex v of G . If G has a proper coloring ϕ such that $\phi(v) \in L(v)$ for all vertices v , then we say that G is L -colorable or ϕ is an L -coloring of G . The graph G is k -choosable if it is L -colorable for every assignment L satisfying $|L(v)| = k$ for all vertices v . The *choice number* or *list chromatic number* $\chi_\ell(G)$ of G is the smallest k such that G is k -choosable. By considering colorings for $E(G)$, we can define analogous notions such as edge assignment, edge- L -coloring, edge- k -choosable, the edge chromatic number $\chi'(G)$, the edge choice number $\chi'_\ell(G)$, etc.

The concept of list coloring was introduced by Vizing [17] and independently by Erdős, Rubin, and Taylor [5]. In recent years, a number of interesting results about the choosability of planar graphs have been obtained, e.g., [2], [6], [14], [15], [16],

*Received by the editors July 25, 2000; accepted for publication (in revised form) June 27, 2002; published electronically September 10, 2002. This research was supported by the Institute of Mathematics, Academia Sinica, Taipei.

<http://www.siam.org/journals/sidma/15-4/37625.html>

[†]Department of Mathematics, Zhejiang Normal University, Jinhua, Zhejiang 321004, People's Republic of China (wangwf0062@sina.com.cn). This work was done while the first author was visiting the Institute of Mathematics, Academia Sinica, Taipei.

[‡]Institute of Mathematics, Academia Sinica, Nankang, Taipei 115, Taiwan (makwlih@sinica.edu.tw).

[18], [19], and [20]. A graph G is said to be k -degenerate if every nonempty subgraph G' of G has a vertex of degree at most k in G' . The list chromatic number of a k -degenerate graph is at most $k + 1$. It is obvious that every planar graph without triangles is 4-choosable because such a graph is 3-degenerate. Lam, Xu, and Liu [12] proved that every planar graph without 4-cycles is 4-choosable. Recently, we have proved in [21] and [22] that every planar graph without 5-cycles or 6-cycles is 4-choosable. These results have been generalized in Lam, Shiu, and Xu [13] to show that, if a planar graph G is free of k -cycles for some $k \in \{3, 4, 5, 6\}$, then G is $(4m, m)$ -choosable for all nonnegative integers m ; i.e., if every vertex is assigned a list of $4m$ colors, then every vertex can be given m colors from its own list so that adjacent vertices get disjoint sets of colors. When $m = 1$, $(4m, m)$ -choosable means the same as 4-choosable.

In this paper, we will prove that every planar graph without intersecting triangles is 4-choosable. The second main result of this paper is to show that the following conjecture holds for a planar graph G without intersecting triangles and $\Delta(G) \neq 5$.

CONJECTURE 1.1. *Every simple graph G is edge- $(\Delta(G) + 1)$ -choosable.*

The above conjecture was first proposed by Vizing (see [11]). An earlier result of Harris [8] asserts that, if G is a graph with $\Delta(G) \geq 3$, then $\chi'_\ell(G) \leq 2\Delta(G) - 2$. This implies Conjecture 1.1 for the case $\Delta(G) = 3$. Recently, Juvan, Mohar, and Škrekovski [10] settled the case for $\Delta(G) = 4$. Conjecture 1.1 has been confirmed for other special cases such as complete graphs [7], graphs with girth at least $8\Delta(G)(\ln \Delta(G) + 1.1)$ [11], and planar graphs with $\Delta(G) \geq 9$ [3].

2. 4-Choosability. To establish our main result about 4-choosability, we need a few lemmas. The first one has already been used in [5].

LEMMA 2.1. *Every cycle of even length is 2-choosable.*

Suppose that L is an assignment for a graph G such that $|L(u)| \geq k(u)$ for each vertex $u \in V(G)$, where $k(u)$ is a nonnegative integer determined by the vertex u . Let L' denote an assignment for G such that $L'(u) \subseteq L(u)$ and $|L'(u)| = k(u)$ for every $u \in V(G)$. Evidently, G is L -colorable if G is L' -colorable. This fact will be used frequently in what follows.

Let A_0 denote a graph obtained by adding a chord u_3u_6 to a 6-cycle $u_1u_2 \cdots u_6u_1$. A graph A is called a *type one* graph if it satisfies the following two conditions.

- (1) A_0 is a spanning subgraph of A .
- (2) A has no intersecting triangles.

The graph A_0 itself is a type one graph.

LEMMA 2.2. *Let A be a type one graph. Let L be an assignment for A such that $|L(u_3)| \geq 2$, $|L(u_6)| \geq 3$, and $|L(u_i)| \geq d_A(u_i)$ for $i = 1, 2, 4, 5$. Then A is L -colorable.*

Proof. It suffices to prove that A is L -colorable when $|L(u_3)| = 2$, $|L(u_6)| = 3$, and $|L(u_i)| = d_A(u_i)$ for $i = 1, 2, 4, 5$.

We note that every edge in $E(A) \setminus E(A_0)$ has one end in $\{u_1, u_2\}$ and the other end in $\{u_4, u_5\}$ by the defining property (2).

If $L(u_1) \cap L(u_3) \neq \emptyset$, we color both u_1 and u_3 with some color $\alpha \in L(u_1) \cap L(u_3)$. Afterwards, we color u_2, u_4, u_5 , and u_6 successively. If $L(u_1) \cap L(u_3) = \emptyset$, then $|(L(u_1) \cup L(u_3)) \setminus L(u_6)| \geq |L(u_1)| + |L(u_3)| - |L(u_6)| \geq 1$. Thus there exists $\beta \in (L(u_1) \cup L(u_3)) \setminus L(u_6)$. When $\beta \in L(u_1)$, we first color u_1 with β then color u_2, u_3, u_4, u_5 , and u_6 successively. When $\beta \in L(u_3)$, we first color u_3 with β then color u_2, u_1, u_4, u_5 , and u_6 successively. \square

Let B_0 be the union of two 4-cycles $C_1 = xu_1u_2u_3x$ and $C_2 = xv_1v_2v_3x$ such that $V(C_1) \cap V(C_2) = \{x\}$. A graph B is called a *type two* graph if it satisfies the following two conditions.

- (1) B_0 is a spanning subgraph of B .
- (2) B has no intersecting triangles.

The graph B_0 itself is a type two graph.

LEMMA 2.3. *Let B be a type two graph. Let L be an assignment for B such that $|L(x)| \geq 3$, $|L(u_i)| \geq d_B(u_i)$, and $|L(v_i)| \geq d_B(v_i)$ for $i = 1, 2, 3$. Then B is L -colorable.*

Proof. It suffices to prove that B is L -colorable when $|L(x)| = 3$, $|L(u_i)| = d_B(u_i)$, and $|L(v_i)| = d_B(v_i)$ for $i = 1, 2, 3$.

We note that every edge in $E(B) \setminus E(B_0)$ has one end in $\{u_1, u_2, u_3\}$ and the other end in $\{v_1, v_2, v_3\}$ by the defining property (2).

If $L(u_1) \cap L(u_3) \neq \emptyset$, we color both u_1 and u_3 with some color $\alpha \in L(u_1) \cap L(u_3)$. Then we color u_2 afterwards. Now each vertex in $V(C_2)$ has at least two colors to choose from. By Lemma 2.1, $V(C_2)$ can be colored properly. If $L(u_1) \cap L(u_3) = \emptyset$, then there exists $\beta \in (L(u_1) \cup L(u_3)) \setminus L(x)$. If $\beta \in L(u_1)$, we first color u_1 with β then color u_2 and u_3 . Again every vertex in $V(C_2)$ has at least two colors to choose from, and hence can be colored properly. The case $\beta \in L(u_3)$ can be treated similarly. \square

Let G be a plane graph with $\delta(G) = 4$. A 4-face $f = [w_1w_2w_3w_4]$ of G is said to be *bad* if $d_G(w_1) = 5$ and $d_G(w_i) = 4$ for $i = 2, 3, 4$. Clearly, every subgraph of a planar graph without intersecting triangles is also a planar graph without intersecting triangles. Every subgraph of a k -choosable (or edge- k -choosable) graph is also k -choosable (or edge- k -choosable). These straightforward facts are tacitly used in the following proofs. Moreover, for a connected plane graph G , the following identity follows from the well-known Euler's formula $|V(G)| - |E(G)| + |F(G)| = 2$.

$$\sum_{v \in V(G)} (2d_G(v) - 6) + \sum_{f \in F(G)} (d_G(f) - 6) = -12. \tag{*}$$

THEOREM 2.4. *If G is a plane graph without intersecting triangles, then G is 4-choosable.*

Proof. Suppose that the theorem is false. Let G be a counterexample plane graph having the fewest vertices. Let L be an arbitrary assignment for G such that $|L(v)| = 4$ for all $v \in V(G)$. Then the following properties (P1) to (P5) hold for G .

(P1) *The minimum degree $\delta(G) \geq 4$.*

If $\delta(G) \leq 3$, let $d_G(u) = \delta(G)$ for some vertex u of G . By the minimality of G , the graph $G - u$ is 4-choosable. Every L -coloring of $G - u$ can be extended to an L -coloring of G since u has at most three forbidden colors. This contradicts the choice of G .

(P2) *The graph G does not contain a 4-face f such that $d_G(v) = 4$ for every boundary vertex v of f .*

Suppose $f = [w_1w_2w_3w_4]$ is a 4-face of G such that $d_G(w_i) = 4$ for $1 \leq i \leq 4$. Since G does not contain intersecting triangles, both w_1w_3 and w_2w_4 are nonedges. Let $H = G - \{w_1, w_2, w_3, w_4\}$. By the minimality of G , the graph H is 4-choosable. Each w_i is adjacent to exactly two vertices in H . By Lemma 2.1, every L -coloring of H can be extended to an L -coloring of G . This contradicts the choice of G .

(P3) *The graph G does not contain two bad 4-faces $f_1 = [u_1u_2u_3u_6]$ and $f_2 = [u_3u_4u_5u_6]$ sharing a unique common edge u_3u_6 and $d_G(u_3) = 5$.*

Suppose on the contrary that G contains such adjacent bad 4-faces f_1 and f_2 . Let $S = \{u_1, u_2, \dots, u_6\}$. The induced subgraph $A = G[S]$ is a type one graph. For any L -coloring of $G - S$, we use U_i to denote the set of colors assigned to the neighbors of u_i in $G - S$. We further let $\sigma(u_i)$ denote the number of those neighbors. On the graph A , define an assignment $L'(u_i) = L(u_i) \setminus U_i$ for $i = 1, 2, \dots, 6$. We note that $|U_i| \leq \sigma(u_i)$, and hence $|L'(u_3)| \geq 2$, $|L'(u_6)| \geq 3$, and $|L'(u_i)| \geq d_G(u_i) - \sigma(u_i) = d_A(u_i)$ for $i \neq 3, 6$. It follows that L' satisfies the conditions of Lemma 2.2, and hence A is L' -colorable. Therefore G is L -colorable, contradicting the choice of G .

(P4) *The graph G does not contain two bad 4-faces $f_1 = [xu_1u_2u_3]$ and $f_2 = [xv_1v_2v_3]$ sharing a unique common vertex x and such that $d_G(x) = 5$.*

Suppose on the contrary that G contains such intersecting bad 4-faces f_1 and f_2 . Let $S = \{x, u_1, u_2, u_3, v_1, v_2, v_3\}$. Thus the induced subgraph $B = G[S]$ is a type two graph. For every L -coloring of $G - S$, we can derive an assignment L' for B such that $|L'(x)| \geq 3$, $|L'(u_i)| \geq d_B(u_i)$, and $|L'(v_i)| \geq d_B(v_i)$ for $i = 1, 2, 3$. It follows from Lemma 2.3 that B is L' -colorable, and hence G is L -colorable. This is a contradiction.

(P5) *A 5-vertex $v \in V(G)$ is incident to at most one bad 4-face.*

Suppose that v is incident to two distinct bad 4-faces $f_1 = [vu_1u_2u_3]$ and $f_2 = [vv_1v_2v_3]$. Then $d_G(u_i) = d_G(v_i) = 4$ for $i = 1, 2, 3$. First we note that $u_1 \neq u_3$ and $v_1 \neq v_3$. Suppose that u_1, u_3, v_1 , and v_3 are arranged around the vertex v in this order. Since $d_G(v) = 5$, it follows that either $u_1 \neq v_3$ or $u_3 \neq v_1$. Without loss of generality, we assume that $u_3 \neq v_1$.

Suppose that $u_1 = v_3$. If $u_2 = v_2$, then $d_G(u_1) = 2$, contradicting (P1). Since G contains no intersecting triangles, $u_2 \neq v_1$ and $u_3 \neq v_2$. This contradicts (P3).

Suppose that $u_1 \neq v_3$. Since G contains no intersecting triangles, $u_2 \neq v_1, v_3$ and $v_2 \neq u_1, u_3$. If, further, $u_2 \neq v_2$, it would contradict (P4).

Finally, suppose that $u_2 = v_2$. Let $S = \{v, u_1, u_2, u_3, v_1, v_3\}$. For every L -coloring of $G - S$, we can trim L to become an assignment L' for the induced subgraph $H = G[S]$ such that $|L'(u_2)| = 4$, $|L'(v)| = 3$, and $|L'(t)| = 2$ for all $t \in \{u_1, u_3, v_1, v_3\}$. If $L'(v) \cap L'(u_2) \neq \emptyset$, we color both v and u_2 with some color $\alpha \in L'(v) \cap L'(u_2)$. Then color u_1, u_3, v_1 , and v_3 . Suppose $L'(v) \cap L'(u_2) = \emptyset$. Then there exist $\beta \in L'(v)$ and $\gamma \in L'(u_2)$ such that $L'(t) \neq \{\beta, \gamma\}$ for $t \in \{u_1, u_3, v_1, v_3\}$. We color v with β , u_2 with γ , and then color t with a color in $L'(t) \setminus \{\beta, \gamma\}$ for $t \in \{u_1, u_3, v_1, v_3\}$. Thus H is always L' -colorable. Hence we arrive at the conclusion that G is L -colorable. This contradiction completes the proof of (P5).

The counterexample graph G is clearly connected. Let c denote the charge function defined on $V(G) \cup F(G)$ by $c(v) = 2d_G(v) - 6$ if $v \in V(G)$ and $c(f) = d_G(f) - 6$ if $f \in F(G)$. Thus $\sum \{c(x) \mid x \in V(G) \cup F(G)\} = -12$ by identity (*). We are going to redistribute the vertex charge $c(v)$ to its incident faces according to the discharging rules (R1)–(R4). During the process, the total sum of all charges is fixed. However, after the discharging is complete, the new charge function $c'(x)$ is nonnegative for all $x \in V(G) \cup F(G)$. This leads to an obvious contradiction.

Now let $v \in V(G)$. By (P1), $d_G(v) \geq 4$ and $c(v) = 2d_G(v) - 6 \geq 2$. The discharging rules are as follows.

(R1) If $d_G(v) \geq 6$, or if $d_G(v) = 5$ and v is incident to neither 3-faces nor bad 4-faces, or if $d_G(v) = 4$ and v is not incident to any 3-faces, then we transfer the amount $w(v)/d_G(v)$ from v to each face in $F(v)$.

(R2) If $d_G(v) = 5$ and v is incident to a 3-face f_1 and a bad 4-face f_2 , then from

v we transfer 1 to each of f_1 and f_2 and $2/3$ to each face in $F(v) \setminus \{f_1, f_2\}$.

(R3) If $d_G(v) = 5$ and v is incident to exactly one face f^* that is a 3-face or a bad 4-face, then from v we transfer 1 to f^* and $3/4$ to each face in $F(v) \setminus \{f^*\}$.

(R4) If $d_G(v) = 4$ and v is incident to a 3-face f_0 , then from v we transfer 1 to f_0 and $1/3$ to each face in $F(v) \setminus \{f_0\}$.

Let $\tau(v \rightarrow f)$ denote the amount transferred out of a vertex v into a face $f \in F(v)$ according to the above rules. Then the following observations are immediate consequences of (R1)–(R4).

Observation 1. For every $v \in V(G)$ and every $f \in F(v)$, $\tau(v \rightarrow f) \geq 1/3$.

Observation 2. For every $v \in V(G)$ with $d_G(v) \geq 6$ and every $f \in F(v)$, $\tau(v \rightarrow f) \geq 1$.

By (P5), the rules imply that at most the amount $c(v)$ is transferred from every vertex v to its incident faces. Thus $c'(v) \geq 0$ for every $v \in V(G)$. It remains to prove that $c'(f) \geq 0$ for every $f \in F(G)$.

If $d_G(f) \geq 6$, then $c'(f) \geq c(f) = d_G(f) - 6 \geq 0$.

If $d_G(f) = 5$, it follows from Observation 1 that $c'(f) \geq c(f) + 5/3 = 2/3$.

If $d_G(f) = 3$, the discharging rules and Observation 2 imply that f receives at least 1 from each of its boundary vertices. So $c'(f) \geq c(f) + 3 = 0$.

If $d_G(f) = 4$, then the boundary of f contains at least one vertex of degree 5 or more by (P2). If the boundary of f contains a vertex v of degree at least 6, then $c'(f) \geq 0$ by Observations 1 and 2. If the boundary of f contains two 5-vertices u_1 and u_2 , it follows from (R1), (R2), and (R3) that $\tau(u_1 \rightarrow f) \geq 2/3$ and $\tau(u_2 \rightarrow f) \geq 2/3$. Hence $c'(f) \geq 0$ by Observation 1. If the boundary of f contains exactly one 5-vertex v , then f is a bad 4-face. By (R2) and (R3), we derive $\tau(v \rightarrow f) = 1$ and $c'(f) \geq 0$. \square

The result of Theorem 2.4 is sharp in the sense that there exist planar graphs G without intersecting triangles such that $\chi_\ell(G) = 4$. Havel [9] gave a planar graph G of order 16 that contains four vertex-disjoint triangles and $\chi(G) = \chi_\ell(G) = 4$. Furthermore, Voigt [19] constructed a triangle-free planar graph G with $\chi(G) = 3$ and $\chi_\ell(G) = 4$. More examples can be found in Aksionov and Mel'nikov [1].

Let T_1 and T_2 be two triangles of a graph G . The *distance* between T_1 and T_2 is the length of a shortest path connecting a vertex of T_1 to a vertex of T_2 . Let $t(G)$ denote the least integer k such that G contains two triangles T_1 and T_2 at distance k . Define $t(G)$ to be infinity if there exists at most one triangle in each component of G . Lam, Shiu, and Xu [13] showed that every planar graph G with $t(G) \geq 2$ is 4-choosable. Theorem 2.4 strengthens this result to prove that $t(G) \geq 1$ guarantees 4-choosability for a planar graph. The condition $t(G) \geq 1$ is essential since there exist non-4-choosable planar graphs G with $t(G) = 0$; see [6, 14, 18]. However, we would like to propose the following conjecture.

CONJECTURE 2.5. *Every planar graph without adjacent triangles is 4-choosable.*

3. Edge- $(\Delta(G) + 1)$ -choosability. Borodin [4] showed that every 3-polytope G , i.e., 3-connected planar graph, without intersecting triangles contains an edge $xy \in E(G)$ such that $d_G(x) + d_G(y) \leq 8$. The following is a generalization of this result.

THEOREM 3.1. *If G is a planar graph without intersecting triangles and $\delta(G) \geq 3$, then there exists an edge $xy \in E(G)$ such that $d_G(x) + d_G(y) \leq 8$.*

Proof. If G is 2-connected, the result follows from Theorem 3.2. Indeed, we may choose any vertex of G as the specific vertex t^* . Otherwise, let H be a block of G that contains a unique cut vertex, say t^* , of G . Since H is 2-connected and $d_H(v) \geq 3$

for all $v \in V(H) \setminus \{t^*\}$, $H - t^*$ contains an edge xy such that $d_H(x) + d_H(y) \leq 8$ by Theorem 3.2. The result follows since $d_G(x) = d_H(x)$ and $d_G(y) = d_H(y)$. \square

THEOREM 3.2. *Let G be a 2-connected plane graph without intersecting 3-faces and let $t^* \in V(G)$ be an arbitrarily chosen vertex. If $d_G(v) \geq 3$ for all $v \in V(G) \setminus \{t^*\}$, then $G - t^*$ contains an edge xy such that $d_G(x) + d_G(y) \leq 8$.*

Proof. To obtain a contradiction, suppose that

$$d_G(x) + d_G(y) \geq 9 \text{ for every edge } xy \text{ of } G - t^*. \tag{**}$$

We again use identity (*) to define the charge function by $c(v) = 2d_G(v) - 6$ if $v \in V(G)$ and $c(f) = d_G(f) - 6$ if $f \in F(G)$. We are going to describe a discharging process that will redistribute the charge $c(v)$ of each vertex v to its incident faces while the total sum of charges is kept fixed. We first note that, since G is 2-connected, the boundary walk of every face of G forms a simple cycle.

(r0) We transfer the amount 2 from t^* to every face in $F(t^*)$.

Let $v \in V(G) \setminus \{t^*\}$. If $d_G(v) = 3$, we do nothing. Otherwise, we redistribute charges according to the following rules.

(r1) If $d_G(v) = 4$, or if $d_G(v) \geq 5$ and v is not incident to any 3-face, then we transfer the amount $c(v)/d_G(v)$ from v to each face in $F(v)$.

(r2) If $d_G(v) \geq 5$ and v is incident to a 3-face f^* with vertices u, v, w , let $d_G(w) \geq d_G(u)$ and consider the following three subcases (r21), (r22), and (r23).

(r21) Assume $d_G(v) = 5$. Then from v we transfer $5/4$ to f^* and $11/16$ to each face in $F(v) \setminus \{f^*\}$.

(r22) Assume $d_G(v) = 6$. We further divide the situation into two subcases:

(r221) If $d_G(u) \geq 4$, let f_1 and f_2 denote the two faces in $F(v) \setminus \{f^*\}$ that are adjacent to f^* . Since G is 2-connected, f_1 is different from f_2 . Then from v we transfer $5/4$ to f^* , $7/8$ to f_1 , $7/8$ to f_2 , and 1 to each face in $F(v) \setminus \{f_1, f_2, f^*\}$.

(r222) If $d_G(u) \leq 3$, let f_{vw} denote the face of G adjacent to f^* such that vw is their common edge. Then from v we transfer $3/2$ to f^* , $1/2$ to f_{vw} , and 1 to each face in $F(v) \setminus \{f^*, f_{vw}\}$.

(r23) If $d_G(v) \geq 7$, then from v we transfer 2 to f^* and $(c(v) - 2)/(d_G(v) - 1)$ to each face in $F(v) \setminus \{f^*\}$.

We still use $\tau(v \rightarrow f)$ to denote the amount transferred out of a vertex v into an incident face f according to the above rules. Then the following claims hold.

Claim 1. If $u \in V(G)$ and $d_G(u) \geq 4$, then $\tau(u \rightarrow f) \geq 1/2$ for all $f \in F(u)$.

Claim 2. If $v \in V(G) \setminus \{t^*\}$ and $d_G(v) \geq 3$, then $\sum\{\tau(v \rightarrow f) \mid f \in F(v)\} \leq c(v)$.

This implies $c'(v) \geq 0$ for all $v \in V(G) \setminus \{t^*\}$ with $d_G(v) \geq 3$.

Claim 3. If $v \in V(G)$ with $d_G(v) \geq 7$, then $\tau(v \rightarrow f) \geq 1$ for all $f \in F(v)$.

Claim 4. Suppose that $f = [u_1u_2u_3u_4]$ is a 4-face of G and t^* is not incident to f . If $d_G(u_1) = d_G(u_3) = 3$, then $\tau(u_2 \rightarrow f) \geq 1$ and $\tau(u_4 \rightarrow f) \geq 1$.

Proof of Claim 4. By (**), we have $d_G(u_2) \geq 6$ and $d_G(u_4) \geq 6$. If $d_G(u_2) \geq 7$, then $\tau(u_2 \rightarrow f) \geq 1$ by Claim 3. So we assume $d_G(u_2) = 6$. If u_2 is not incident to any 3-face, then $\tau(u_2 \rightarrow f) = 1$ by (r1). Otherwise, we suppose that u_2 is incident to a 3-face $f_0 = [u_2xy]$. If f_0 is not adjacent to f , then $\tau(u_2 \rightarrow f) = 1$ by (r221) and (r222). If f_0 is adjacent to f , we may suppose that $x = u_1$. Thus $d_G(y) \geq 6$. It is easy to see that $\tau(u_2 \rightarrow f) = 1$ by (r222). Similarly, we can prove that $\tau(u_4 \rightarrow f) \geq 1$. This completes the proof of Claim 4.

Now let us verify $c'(f) \geq 0$ for all $f \in F(G)$. If $d_G(f) \geq 6$, then $c'(f) \geq c(f) = d_G(f) - 6 \geq 0$. Otherwise, we consider two cases as follows.

Assume that t^* is incident to f . If $4 \leq d_G(f) \leq 5$, then $c'(f) \geq c(f) + 2 \geq -2 + 2 = 0$ by (r0). Assume $f = [xyt^*]$ is a 3-face with $d_G(x) \leq d_G(y)$. If $d_G(x) \geq 4$, then $c'(f) \geq -3 + 2 + 1/2 + 1/2 = 0$ by (r0) and Claim 1. If $d_G(x) = 3$, (**) guarantees $d_G(y) \geq 6$, and hence $c'(f) \geq -3 + 2 + \frac{3}{2} = 1/2$ by (r0), (r222), and (r23).

Next assume that t^* is not incident to f . If $d_G(f) = 5$, it follows from (**) that the boundary of f contains at least three vertices of degree ≥ 5 . By Claim 1, $c'(f) \geq c(f) + 3/2 = 1/2$.

If $d_G(f) = 3$, we let $f = [vuw]$ and suppose that $d_G(w) \geq d_G(u) \geq d_G(v)$. If $d_G(v) \geq 5$, then $c'(f) = c(f) + \tau(v \rightarrow f) + \tau(u \rightarrow f) + \tau(w \rightarrow f) \geq -3 + \frac{5}{4} + \frac{5}{4} + \frac{5}{4} = 3/4$ by (r2). If $d_G(v) = 4$, then (**) implies that $d_G(w) \geq d_G(u) \geq 5$. Thus $c'(f) \geq 0$ by (r1) and (r2). If $d_G(v) = 3$, then $d_G(w) \geq d_G(u) \geq 6$. It follows from (r222) and (r23) that $c'(f) \geq 0$.

Finally, we suppose that $d_G(f) = 4$, and hence $c(f) = -2$. Let $f = [u_1u_2u_3u_4]$ satisfying $d_G(u_1) = \min\{d_G(u_i) \mid 1 \leq i \leq 4\}$. If $d_G(u_1) \geq 4$, then $c'(f) \geq c(f) + 2 = 0$ by Claim 1. So we assume $d_G(u_1) = 3$. It follows that $d_G(u_2) \geq 6$ and $d_G(u_4) \geq 6$. Assume $d_G(u_4) \geq d_G(u_2)$. Our proof is further divided into three subcases.

(C1) $d_G(u_2) \geq 7$. Then Claim 3 guarantees $c'(f) \geq 0$.

(C2) $d_G(u_4) \geq 7$ and $d_G(u_2) = 6$. If $d_G(u_3) \geq 4$, then by Claims 1 and 3 we have $c'(f) \geq 0$. If $d_G(u_3) = 3$, then $\tau(u_2 \rightarrow f) \geq 1$ and $\tau(u_4 \rightarrow f) \geq 1$ by Claim 4, and hence $c'(f) \geq 0$.

(C3) $d_G(u_2) = d_G(u_4) = 6$. If $d_G(u_3) = 3$, we have $c'(f) \geq 0$ by Claim 4. If $d_G(u_3) \geq 7$, then $\tau(u_3 \rightarrow f) \geq 1$. By Claim 1, $c'(f) \geq 0$. Now assume $4 \leq d_G(u_3) \leq 6$. If u_2u_3 is not in the boundary of any 3-face, then $\tau(u_2 \rightarrow f) = 1$ by (r1), (r221), or (r222) (with $u = u_1$ in (r222)). By Claim 1, we obtain $c'(f) \geq 0$. If u_2u_3 is in the boundary of a 3-face, then u_3u_4 is not in the boundary of a 3-face since G has no intersecting 3-faces, and so $c'(f) \geq 0$ in this case also.

It follows that $c'(x) \geq 0$ for all $x \in (V(G) \cup F(G)) \setminus \{t^*\}$. However, $c'(t^*) = c(t^*) - 2|F(t^*)| \geq 2d_G(t^*) - 6 - 2d_G(t^*) = -6$. Thus we obtain the following contradiction.

$$-12 = \sum \{c(x) \mid x \in V(G) \cup F(G)\} = \sum \{c'(x) \mid x \in V(G) \cup F(G)\} \geq -6. \quad \square$$

The following is a direct consequence of Theorem 3.1.

COROLLARY 3.3. *If G is a planar graph without intersecting triangles, then $\delta(G) \leq 4$.*

We note that the upper bound in Theorem 3.1 is tight. We can construct infinitely many examples of 4-regular plane graphs without intersecting triangles. For instance, we take any 3-regular triangle-free 3-polytope (e.g., the dual of any 4-connected triangulation) and replace every edge by two (geometrically) parallel edges and every vertex by three vertices forming a triangle.

THEOREM 3.4. *Every planar graph G without intersecting triangles is edge- k -choosable, where $k = \max\{7, \Delta(G) + 1\}$.*

Proof. The proof is carried out by induction on the number $|E(G)|$ of edges. The theorem holds trivially when $|E(G)| \leq 3$. Let G be a planar graph without intersecting triangles such that $|E(G)| \geq 4$. Suppose that L is an edge assignment for G such that $|L(e)| = k$ for each $e \in E(G)$. Let $\delta'(G) = \min\{d_G(v) \mid d_G(v) > 0\}$. If $\delta'(G) \leq 2$, let e be an edge of G incident to a vertex of degree $\delta'(G)$ in G . By the induction hypothesis, $G - e$ has an edge- L -coloring ϕ . We can extend ϕ to the edge e because there are at most $\Delta(G)$ forbidden colors for e , whereas the number of available colors is at least $\Delta(G) + 1$. If $\delta'(G) \geq 3$, G contains an edge xy such that $d_G(x) + d_G(y) \leq 8$ by Theorem 3.1. The induction hypothesis implies that $G - xy$

has an edge- L -coloring ϕ . We can color xy with some color from $L(xy)$ that was not used by ϕ on the edges adjacent to xy . Since there exist at most six such edges and $|L(xy)| = k \geq 7$, the required color is available. \square

Combining Theorem 3.4 and the results of [8] and [10], we obtain the following.

THEOREM 3.5. *If G is a planar graph without intersecting triangles and $\Delta(G) \neq 5$, then $\chi'_\ell(G) \leq \Delta(G) + 1$.*

Acknowledgment. The authors would like to thank the referees for valuable suggestions to improve this work. The examples supplied after Corollary 3.3 are due to one of the anonymous referees.

REFERENCES

- [1] V. A. AKSIONOV AND L. S. MEL'NIKOV, *Some counterexamples associated with the three-color problem*, J. Combin. Theory Ser. B, 28 (1980), pp. 1–9.
- [2] N. ALON AND M. TARSI, *Colorings and orientations of graphs*, Combinatorica, 12 (1992), pp. 125–134.
- [3] O. V. BORODIN, *A generalization of Kotzig's theorem and prescribed edge coloring of planar graphs*, Mat. Zametki, 48 (1990), pp. 22–28, 160 (in Russian).
- [4] O. V. BORODIN, *An extension of Kotzig's theorem on the minimum weight of edges in 3-polytopes*, Math. Slovaca, 42 (1992), pp. 385–389.
- [5] P. ERDŐS, A. L. RUBIN, AND H. TAYLOR, *Choosability in graphs*, Congr. Numer., 26 (1980), pp. 125–157.
- [6] S. GUTNER, *The complexity of planar graph choosability*, Discrete Math., 159 (1996), pp. 119–130.
- [7] R. HÄGGKVIST AND J. JANSSEN, *New bounds on the list-chromatic index of the complete graph and other simple graphs*, Combin. Probab. Comput., 6 (1997), pp. 295–313.
- [8] A. J. HARRIS, *Problems and Conjectures in Extremal Graph Theory*, Ph.D. dissertation, Cambridge University, UK, 1984.
- [9] I. HAVEL, *On a conjecture of B. Grünbaum*, J. Combinatorial Theory, 7 (1969), pp. 184–186.
- [10] M. JUVAN, B. MOHAR, AND R. ŠKREKOVSKI, *Graphs of degree 4 are 5-edge-choosable*, J. Graph Theory, 32 (1999), pp. 250–262.
- [11] A. V. KOSTOCHKA, *List edge chromatic number of graphs with large girth*, Discrete Math., 101 (1992), pp. 189–201.
- [12] B. C. P. LAM, B. XU, AND J. LIU, *The 4-choosability of plane graphs without 4-cycles*, J. Combin. Theory Ser. B, 76 (1999), pp. 117–126.
- [13] B. C. P. LAM, W. C. SHIU, AND B. XU, *On structure of some plane graphs with applications to choosability*, J. Combin. Theory Ser. B, 82 (2001), pp. 285–296.
- [14] M. MIRZAKHANI, *A small non-4 choosable planar graph*, Bull. Inst. Combin. Appl., 17 (1996), pp. 15–18.
- [15] C. THOMASSEN, *Every planar graph is 5-choosable*, J. Combin. Theory Ser. B, 62 (1994), pp. 180–181.
- [16] C. THOMASSEN, *3-list coloring planar graphs of girth 5*, J. Combin. Theory Ser. B, 64 (1995), pp. 101–107.
- [17] V. G. VIZING, *Coloring the vertices of a graph in prescribed colors*, Diskret. Analiz, 29 (1976), pp. 3–10, 101 (in Russian).
- [18] M. VOIGT, *List colorings of planar graphs*, Discrete Math., 120 (1993), pp. 215–219.
- [19] M. VOIGT, *A not 3-choosable planar graph without 3-cycles*, Discrete Math., 146 (1995), pp. 325–328.
- [20] M. VOIGT AND B. WIRTH, *On 3-colorable non-4-choosable planar graphs*, J. Graph Theory, 24 (1997), pp. 233–235.
- [21] W. WEIFAN AND KO-WEI LIH, *The 4-choosability of planar graphs without 6-cycles*, Australas. J. Combin., 24 (2001), pp. 157–164.
- [22] W. WEIFAN AND KO-WEI LIH, *Choosability and edge choosability of planar graphs without five cycles*, Appl. Math. Lett., 15 (2002), pp. 561–565.

A FAST ALGORITHM FOR GENERATING NONISOMORPHIC CHORD DIAGRAMS*

JOE SAWADA[†]

Abstract. Using a new string representation, we develop two algorithms for generating nonisomorphic chord diagrams. Experimental evidence indicates that the latter of the two algorithms runs in constant amortized time. In addition, we use simple counting techniques to derive a formula for the number of nonisomorphic chord diagrams.

Key words. chord diagram, generation algorithm, enumeration, necklace

AMS subject classifications. 05-04, 05A15, 68R05, 68R15

PII. S0895480100377970

1. Introduction. Chord diagrams are the fundamental combinatorial objects underlying Vassiliev invariants, which have applications in knot theory [1]. A *chord diagram* is a set of $2n$ points on an oriented circle (counterclockwise) joined pairwise by n chords. Figure 1 illustrates a chord diagram with four chords. Two chord diagrams are isomorphic if one can be obtained by some rotation of the other. Special instances of chord diagrams are shown to have application in stamp foldings by Koehler [7]. A related object called a linearized chord diagram is studied by Stoimenow in [12] and braided chord diagrams are discussed by Birman and Trapp in [2].

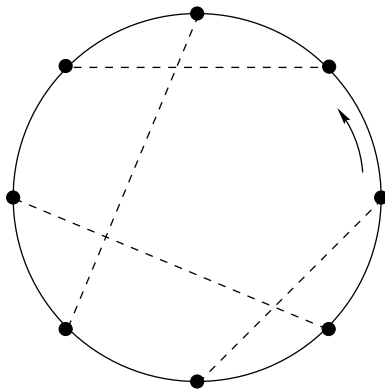


FIG. 1. *Chord diagram with four chords.*

Two fundamental questions when dealing with any combinatorial object are the following:

1. How many instances of the object are there? (i.e., How many nonisomorphic chord diagrams are there with n chords?)

*Received by the editors September 12, 2000; accepted for publication (in revised form) June 27, 2002; published electronically September 10, 2002. This research was supported by NSERC and in part by Czech grant GAČR 201/99/0242 and ITI under project LN-00A 056.

<http://www.siam.org/journals/sidma/15-4/37797.html>

[†]Department of Computer Science, University of Toronto, Toronto, ON, Canada (jsawada@cs.toronto.edu).

2. How can we efficiently generate (list) all instances of the object? (i.e., Can we develop a fast algorithm to generate all nonisomorphic chord diagrams with n chords?)

In response to the first question, three independent papers by Li and Sun [8], Cori and Marcus [4], and Stoimenow [13] have derived enumeration formulas for the number of nonisomorphic chord diagrams. In each of these papers, the exact formula is the main result; however, in each case the derivation of the formula uses relatively complex methods. Cori and Marcus use Burnside’s lemma (stated in section 3) along with liftings of quasidiagrams; Li and Sun introduce a new object called a generalized m -configuration; Stoimenow uses Burnside’s lemma along with two new objects: linearized chord diagrams and generalized linearized chord diagrams. As a secondary result in this paper, we derive an exact formula for the number of nonisomorphic chord diagrams with n chords using simple counting techniques.

The second question has not received as much attention as the first, or at least no significant results have been previously recorded. In response to this open problem, we develop two algorithms for generating nonisomorphic chord diagrams using a new string representation. A primary goal in any generation algorithm is for the amount of computation to be proportional to the number of objects generated. Such algorithms are said to be CAT, for constant amortized time. The first algorithm we develop is very simple but does not attain this time bound. The second algorithm requires more explanation; however, experimental evidence gives a strong indication that it is CAT.

In the following section we give some basic number theory definitions, along with a background of a related object called a necklace. In section 3 we derive an exact formula for enumerating nonisomorphic chord diagrams using simple techniques. In section 4 we describe a new string representation for chord diagrams. Then, in section 5, we outline a simple generation algorithm for nonisomorphic chord diagrams. In section 6 we present another generation algorithm, with experimental results indicating that the algorithm is CAT. We conclude with a discussion of future work and open problems in section 7.

2. Background. In the next section we derive an exact formula for the number of nonisomorphic chord diagrams with n chords. In the derivation, we encounter the following number theoretic functions.

The *Euler totient* function on an integer n , denoted $\phi(n)$, is the number of positive integers less than n that are relatively prime to n .

The bifactorial of an integer n , denoted $n!!$, is defined by the following:

$$n!! = \begin{cases} \prod_{j=0}^{\lfloor \frac{n-1}{2} \rfloor} n - 2j & \text{if } n > 0, \\ 1 & \text{if } n = 0 \text{ or } n = -1, \\ 0 & \text{if } n \leq -2. \end{cases}$$

Using this notation, it is easy to see that the number of chord diagrams with n chords is $(2n - 1)!!$.

2.1. Necklaces. An object closely related to a chord diagram is a necklace. A *necklace* is the lexicographically smallest element of an equivalence class of k -ary strings under rotation. For example, the set of all binary necklaces of length 4 is $\{0000, 0001, 0011, 0101, 0111, 1111\}$. We call an aperiodic necklace a *Lyndon word* and a string that is a prefix of a necklace a *prenecklace*. We will reserve the term *periodic necklace* for a necklace that is not a Lyndon word.

```

procedure GenNecklaces (  $t, p$  : integer );
local  $j$  : integer;
begin
  if  $t > n$  then
    if  $n \bmod p = 0$  then Print()
  else begin
    for  $j \in \{a_{t-p}, \dots, k-2, k-1\}$  do begin
       $a_t := j$ ;
      if  $a_t = a_{t-p}$  then GenNecklaces(  $t+1, p$  );
      else GenNecklaces(  $t+1, t$  );
    end;
  end;
end;

```

FIG. 2. *The recursive necklace generation algorithm.*

Later, when we outline two algorithms for generating nonisomorphic chord diagrams, we follow the methods used in Ruskey's recursive necklace generation algorithm $\text{GenNecklaces}(t, p)$ shown in Figure 2 [3]. This algorithm has been the basis for generating many other objects with rotational equivalence. In particular, it has been used to develop CAT algorithms to generate bracelets [11], fixed density necklaces [9], and unlabeled necklaces [3]. The general idea of this backtracking algorithm is to generate a length t prenecklace, stored in the array a , and then for each valid character append it to the end of the prenecklace to get a length $t+1$ prenecklace. The parameter p maintains the length of the longest Lyndon prefix of the string. When the prenecklace is of length n , a simple test determines whether or not it is a necklace. This algorithm can also generate Lyndon words by changing the condition from $n \bmod p = 0$ to $n = p$. The initial call is $\text{GenNecklaces}(1, 1)$, and a_0 is initially set to 0. The function $\text{Print}()$ prints out the string $a_1 a_2 \cdots a_n$. A more detailed explanation and a proof showing the algorithm is CAT is found in [3].

3. Enumerating nonisomorphic chord diagrams. One of the most useful tools for enumerating combinatorial objects with equivalence under some group action is Burnside's lemma.

BURNSIDE'S LEMMA. *If a group G acts on a set S and $\text{Fix}(g) = \{s \in S \mid g(s) = s\}$, then the number of equivalence classes is given by*

$$\frac{1}{|G|} \sum_{g \in G} |\text{Fix}(g)|.$$

The set of all chord diagrams with n chords is partitioned into equivalence classes by the cyclic group \mathbb{C}_{2n} . Two chord diagrams are isomorphic if one can be obtained by some rotation of the other. If we let σ denote a single rotation by $(360/2n)$ degrees, then the group elements of \mathbb{C}_{2n} are σ^j for $j = 1, 2, \dots, 2n$. To count the number of nonisomorphic chord diagrams with n chords, which we denote $C(n)$, we apply Burnside's lemma:

$$C(n) = \frac{1}{2n} \sum_{j=1}^{2n} \text{Fix}(\sigma^j).$$

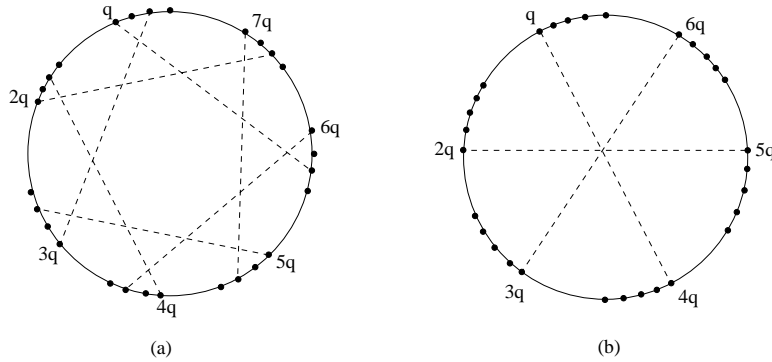


FIG. 3. (a) One of the $2n - p$ possible lengths for the chords starting at $q, 2q, \dots, pq$. (b) For p even, there is only one choice for the endpoint landing back in the list $q, 2q, \dots, pq$.

The number of chord diagrams fixed by σ^j depends only on the order of σ^j . In other words, if two group elements σ^j and σ^k have the same order, then the set of chord diagrams fixed by each group element will be the same. The number of elements of \mathbb{C}_{2n} with order p (where $p|2n$) is $\phi(p)$. Thus, if we let $T(2n, p)$ denote the number of chord diagrams with n chords fixed by a group element of order p (namely σ^q), then

$$C(n) = \frac{1}{2n} \sum_{pq=2n} \phi(p)T(2n, p).$$

We now derive a formula for $T(2n, p)$ by deriving recurrence equations for two cases: p odd and p even. We start by labeling the endpoints on a chord diagram from 1 to $2n$ in counterclockwise order around the circle. For each endpoint i we consider the chord that touches i to start at i and end at its other endpoint j . With this labeling, we define the *length* of a chord starting from i and ending at j to be $(j - i) \bmod 2n$. We now consider the chords starting at $q, 2q, \dots, pq$, where $pq = 2n$. If a chord diagram is fixed by σ^q , then the length of the chords starting at these positions must be the same. If p is odd, then there are $2n - p$ possible lengths for the chords, since it is impossible for two endpoints in the list $q, 2q, \dots, pq$ to be joined together (see Figure 3(a)). If we now ignore these chords and their $2p$ endpoints, we are reduced to the problem of counting $T(2n - 2p, p)$. Thus, if p is odd,

$$T(2n, p) = (2n - p)T(2n - 2p, p).$$

In the base case, $T(0, p) = 1$. If p is even, then there is also one way for the chords to have both endpoints in the set $q, 2q, \dots, pq$. This case arises when there are $p/2$ chords of length n which means that there are only p endpoints to ignore (see Figure 3(b)). Therefore, if p is even,

$$T(2n, p) = (2n - p)T(2n - 2p, p) + T(2n - p, p).$$

In the base cases, $T(p, p) = T(0, p) = 1$.

Solving the two recurrence equations yields the following exact formula:

$$T(2n, p) = \begin{cases} p^{\frac{p}{2}}(q - 1)!! & \text{if } p \text{ odd,} \\ \sum_{j=0}^{\lfloor \frac{p}{2} \rfloor} p^j \binom{q}{2j} (2j - 1)!! & \text{if } p \text{ even.} \end{cases}$$

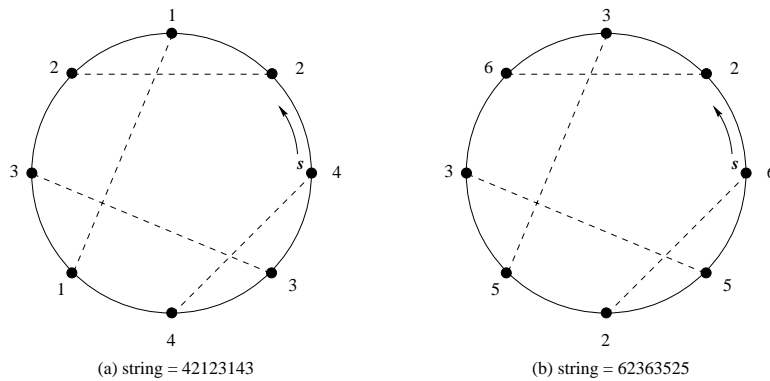


FIG. 4. *Two string representations: (a) label chords then endpoints; (b) label endpoints by chord length.*

The solution for odd p is easily obtained by substituting into the recurrence. Proof by induction on q will verify the solution when p is even.

4. Representing chord diagrams. There are numerous ways to represent chord diagrams. Several objects equivalent to our definition of chord diagrams have been studied by other authors, including polygons where the sides are identified pairwise [14, 15] and one-vertex maps [6]. In this section we develop a new string representation.

Before we describe this new string representation for chord diagrams, we outline a very natural one. First, assign each chord a unique value from 1 to n and then label the endpoints with the value of their incident chord. If we arbitrarily pick a starting point s , then we obtain a string representation by recording the endpoint values starting at s and moving counterclockwise (by convention) around the circle. In this manner, any string with length $2n$ containing exactly two occurrences of the values 1 through n can be used to represent a chord diagram. An example of this string representation is shown in Figure 4(a). Such string representations have equivalence under string rotation and permutation of the alphabet symbols 1 through n . Thus, there may be up to $2n(n!)$ strings in each equivalence class. The lexicographically smallest strings in each equivalence class are more commonly known as *unlabeled necklaces* (where the number of each alphabet symbol is 2). Currently, there exists an efficient algorithm for generating binary unlabeled necklaces [3]; however, no efficient algorithm exists for strings on an arbitrarily sized alphabet. There also exists an efficient algorithm to generate necklaces where the number of 0's is fixed [9], but there is currently no efficient algorithm to generate necklaces if the number of each alphabet symbol is fixed.

Because no efficient generation algorithm currently exists using this natural string representation, we consider a new approach. This time we label each endpoint with its associated length (see section 3). Note that the lengths are independent of the starting point s ; however, there is a dependency between each pair of endpoints joined by a chord—their values must sum to $2n$. If we again traverse counterclockwise around the circle starting at s , recording the endpoint values, we obtain a new string representation. In this new string representation, we no longer have equivalence under permutation of the alphabet symbols, and the number of each alphabet symbol is no longer fixed; however, the size of the alphabet has increased from n to $2n - 1$. An

example of this string representation is given in Figure 4(b).

In each of the following two sections, we present an algorithm for generating nonisomorphic chord diagrams. Both algorithms use the new string representation outlined in this section; however, each algorithm defines a different representative for each equivalence class.

5. A simple algorithm. In this section we develop a simple algorithm to list all nonisomorphic chord diagrams with n chords. To represent the chord diagrams, we use the new string representation described in the previous section. Using the lexicographically smallest string as the representative of each equivalence class, we arrive at a problem equivalent to generating length $2n$ necklaces on an alphabet of size $2n - 1$, with the added restriction that each necklace corresponds to a valid chord diagram.

Recall that when generating necklaces, we build up a prenecklace one character at a time. Applying this to chord diagrams, we instead add one chord or two characters at a time. Thus, if we are adding the value j to the t th position of the string, then we must also add the value $2n - j$ to the $(t + j)$ th position. Of course, we must observe the condition that $t + j \leq 2n$. In addition, we must make sure that we do not overwrite values already assigned to positions t and $t + j$ in the prenecklace. If we have already assigned a value to the t th position (i.e., if $a_t \neq 0$), then we continue generation with position $t + 1$ only if the string $a_1 \dots a_t$ is a valid prenecklace (i.e., $a_t \geq a_{t-p}$). If $a_t < a_{t-p}$, then any chord diagram with prefix $a_1 a_2 \dots a_t$ will not be lexicographically minimal under rotation [3]. By adding these simple modifications to $\text{GenNecklaces}(t, p)$, we ensure that each necklace generated corresponds to a valid chord diagram. The resulting algorithm for generating nonisomorphic chord diagrams in lexicographic order, $\text{SimpleChords}(t, p)$, is shown in Figure 5. The initial call is $\text{SimpleChords}(1, 1)$, and a_0 is initially set to 1. The function $\text{Print}()$ prints out the string $a_1 a_2 \dots a_{2n}$. Aperiodic chord diagrams can be generated by replacing the test $2n \bmod p = 0$ with $2n = p$, as was the case with necklaces.

Recall that our goal is to develop a generation algorithm which runs in constant amortized time. The goal does not look promising with this algorithm since the depth of the computation tree is $2n$ when we require only the assignment of n chords per diagram. To verify this conjecture we gather some experimental evidence. To calculate the amount of computation we sum the number of recursive calls plus the number of iterations of the **for** loop that did not produce a recursive call. The resulting ratio of this computation compared to the number of chord diagrams generated is given in Table 1 for $n \leq 11$. Notice that the ratios are steadily increasing as the number of chords increases. This is a strong indication that the algorithm is *not* CAT. For this reason, we attempt no mathematical analysis and focus on developing a faster algorithm.

6. A fast algorithm. In this section we develop an experimentally CAT algorithm for generating nonisomorphic chord diagrams. In this algorithm we use the same string representation for chord diagrams as in the previous algorithm, but this time we use a different representative for each equivalence class.

Let $\alpha = a_0 a_1 a_2 \dots a_{2n-1}$ represent a chord diagram with n chords. Let pos_i be the increasing sequence (possibly empty) composed of the positions (indexes) for all occurrences of the value i in α . Now consider the string $\beta = pos_1 pos_2 pos_3 \dots pos_{2n-1}$. Using this construction, each string α yields a unique string β . We define the canonical form, or representative, of each equivalence class to be the string α with the lexico-

```

procedure SimpleChords (  $t, p$  : integer );
local  $j$  : integer;
begin
  if  $t > 2n$  then
    if  $2n \bmod p = 0$  then Print()
  else begin
    if  $a_t = 0$  and  $t + a_{t-p} \leq 2n$  then begin
      for  $j \in \{a_{t-p}, \dots, 2n - t\}$  do begin
        if  $a_{t+j} = 0$  then begin
           $a_t := j$ ;  $a_{t+j} := 2n - j$ ;
          if  $a_t = a_{t-p}$  then SimpleChords(  $t + 1, p$  );
          else SimpleChords(  $t + 1, t$  );
           $a_{t+j} := 0$ ;
        end;
      end;
       $a_t := 0$ ;
    end;
    else if  $a_t = a_{t-p}$  then SimpleChords(  $t + 1, p$  );
    else if  $a_t > a_{t-p}$  then SimpleChords(  $t + 1, t$  );
  end;
end;

```

FIG. 5. A simple algorithm for generating nonisomorphic chord diagrams with n chords.

TABLE 1
Experimental results for *SimpleChords*(t, p).

Number of chords n	Nonisomorphic chord diagrams	Ratio of work done to chord diagrams generated
1	1	1.0
2	2	3.0
3	5	8.0
4	18	11.8
5	105	14.3
6	902	15.7
7	9749	16.9
8	127072	17.9
9	1915951	18.8
10	32743182	19.8
11	624999093	20.7

graphically smallest string β . For example, in Table 2 we show the equivalence class of strings representing the chord diagram in Figure 4(b) along with their corresponding β strings.

Before we develop a generation algorithm using these representatives, we first outline a linear time verification algorithm for determining whether or not the string α (representing a chord diagram) is in canonical form.

6.1. A verification algorithm. A naïve method for determining if a chord diagram α is in canonical form is to compare its β string with the β string of all other strings in its equivalence class. Such an algorithm would take worst case time $O(n^2)$. We present an algorithm that runs in linear time.

By the definition of the canonical form, we see that the positions of the minimum

TABLE 2
 The canonical form for this equivalence class is 25623635.

α	β
62363525	16245703
23635256	05134627
36352562	47023516
63525623	36172405
35256236	25061347
52562363	14570236
25623635	03461725
56236352	27350614

value in the string $\alpha = a_0a_1 \cdots a_{2n-1}$ are the most critical. If v^* is the minimum value, then we consider the string $pos_{v^*} = p_1p_2 \cdots p_t$, where there are t occurrences of the value v^* in α . In order for α to be in canonical form then p_1 must equal 0 or, equivalently, $a_0 = v^*$. If p_1 had any other value, then there would exist a rotation of α such that $p_1 = 0$. This would yield a smaller pos_{v^*} string, and thus a smaller β string. Now consider the modified string $pos'_{v^*} = q_1q_2 \cdots q_t$, where $q_i = p_{i+1} - p_i$ for $i = 1, 2, \dots, t - 1$ and $q_t = 2n - p_t$. If the string pos'_{v^*} is a necklace, then it is easy to verify that the original string pos_{v^*} will be the lexicographically smallest string when compared to the corresponding pos_{v^*} strings from other strings in α 's equivalence class. Furthermore, if pos'_{v^*} is a Lyndon word, then α will be the unique string in its equivalence class to yield the string pos_{v^*} , and thus it is in canonical form. If pos'_{v^*} is not a necklace, then we can find a rotation of the string α such that a smaller string pos_{v^*} can be obtained, implying that α is *not* in canonical form. As an example to the above strategy, consider the string $\alpha = 363959463789$. Since the minimum value is 3, we consider $pos_3 = 028$ and $pos'_3 = 264$. Because pos'_3 is a Lyndon word, α is in canonical form.

Using this strategy, we can determine whether or not a string α is in canonical form unless the string pos'_{v^*} is a periodic necklace. If pos'_{v^*} has length t and period p , assign $p' = 2n(\frac{p}{t})$; then the rotations of the string α starting at positions $p', 2p', \dots, 2n - p'$ will all yield the same string pos_{v^*} . In this case, we must continue examining α 's corresponding β string. We update the value v to the next smallest value found in α and focus on the new string pos_v . Observe that we can no longer employ the same strategy as before, since the starting points for the other rotations of α that may be the canonical form have been restricted. Of these remaining strings, for α to be in canonical form, it must have the lexicographically smallest string pos_v . To determine this efficiently, we modify the string pos_v in the following manner. First, the values $p', 2p', \dots, 2n - p', 2n$ are inserted into pos_v so the string is still in sorted order. Then each value j is replaced with $j \bmod p'$. Finally, we replace all 0's, which were originally the values $p', 2p', \dots, 2n$, with p' . We denote this modified string by pos'_v . Notice that such a construction implies that the string pos'_v for $\sigma^j(\alpha)$, where j is $p', 2p', \dots, 2n - p'$, is a rotation of the string pos'_v for α . Thus, as before, if the resulting string pos'_v is a Lyndon word, then α is in canonical form. If pos'_v is a periodic necklace with period p and length t , then we repeat this procedure with the next largest v , updating p' to $2n(\frac{p}{t})$. If pos'_v is not a necklace, then α is not in canonical form. Due to the dependencies on the string α , if v ever exceeds n , then the chord diagram is in canonical form and has period equal to the last updated value for p' .

To get a better understanding of this verification algorithm, we go through two examples.

Example 1. Consider a chord diagram represented by $\alpha = 1925819258$. We want to determine if α is in canonical form. First we consider $pos_1 = 05$ and $pos'_1 = 55$. Since pos'_1 is a periodic necklace we must consider $pos_2 = 27$, with $p' = 5$. To modify pos_2 , we insert the value 5 and 10 to get the string 2 5 7 10. Next, we replace each value j with $j \bmod 5$ to get 2020. Finally, we replace the 0's with 5 to get the new string $pos'_2 = 2525$. Since this is a periodic necklace, we must repeat this procedure for the string pos_5 updating $p' = 5$. We now consider $pos_5 = 38$ and perform the modifications to get $pos'_5 = 3535$. Again we have a periodic necklace and update $p' = 5$. Since the next value exceeds n , we conclude that α is in canonical form (with period 5). \square

Example 2. Consider the string $\alpha = 3\ 6\ 10\ 13\ 11\ 4\ 7\ 10\ 3\ 12\ 4\ 13\ 6\ 9\ 12\ 5$ representing a chord diagram with eight chords. To determine if it is in canonical form we first consider $pos_3 = 08$ with $pos'_3 = 88$. Since the latter string is a periodic necklace we must consider $pos_4 = 5\ 10$ with $pos'_4 = 5828$. Now since 5828 is not a necklace, the string α is *not* in canonical form. \square

In the worst case, this verification algorithm must analyze each string pos'_v for $v = 1, 2, \dots, n$. Using Duval's algorithm for factoring a string into Lyndon words [5], we can determine if pos'_v is a necklace or a Lyndon word in linear time. Therefore, an upper bound for the running time of the algorithm is proportional to $\sum_{v=1}^n |pos'_v|$. Observe that length of each string pos'_v is at most $|pos_v| + |pos_{v-1}|$. Thus, since $\sum_{v=1}^n |pos_v| \leq 2n$, the verification algorithm runs in time $O(n)$.

6.2. The generation algorithm. In this subsection we describe a fast algorithm for generating chord diagrams. The method behind the generation algorithm follows directly from the verification algorithm described in the previous subsection.

Following the verification algorithm, the placement of the minimum value v^* is the most important. Specifically, the value v^* must occur in the position a_0 , and the string pos'_{v^*} must be a necklace. Thus, the first step in the generation algorithm is to generate all strings pos_{v^*} (the placing of the values v^* in α) so that the corresponding string pos'_{v^*} is a necklace. For each string pos'_{v^*} that is a Lyndon word, it does not matter how the rest of the string α is filled as long as each position has value at least $v^* + 1$. Of course, each value added to a string represents an endpoint of a chord whose other endpoint must be added simultaneously, so that whenever the value v is added to position s the value $2n - v$ must be added to position $(s + v) \bmod 2n$. If the string pos'_{v^*} is a periodic necklace, then we repeat the process by attempting to place the next largest value v in such a way that pos'_v is a necklace. The result of this approach is the generation of all strings α which represent unique chord diagrams.

This algorithm is naturally divided into three separate recursive routines: the first routine $\text{Gen}(t, p, s, v^*, \text{last}, B)$ generates the necklaces pos'_{v^*} ; the second routine $\text{Gen2}(t, p, s, v, p', \text{part})$ generates the necklaces pos'_v for all $v > v^*$; and the third routine $\text{GenRest}(s, e, v)$ fills the remaining positions with values that are at least v . The routine $\text{FastChords}()$ drives these routines to generate all nonisomorphic chord diagrams with n chords.

Within the algorithm a global linked list is used to keep track of the available positions of α , in increasing order. The variable head is the value of the first available position, and the value $2n$ represents the end of the list. If s is an available position in the list, then $s.\text{next}$ will give the value of the next available position in the list. If the list is implemented using an array with next and previous pointers, then the functions $\text{Add}(s)$, $\text{Remove}(s)$, and $\text{Avail}(s)$ can be implemented in constant time. The boolean function $\text{Avail}(s)$ returns TRUE if s is in the list of available positions and FALSE


```

procedure FastChords ();
local  $i, v^*$  : integer;
begin
  InitList();
  for  $v^* \in \{1, 2, \dots, n - 1\}$  do  $pos_{v^*,0} := 0$ ;
  for  $v^* \in \{1, 2, \dots, n - 1\}$  do begin
     $a_0 := v^*$ ;  $a_{v^*} := 2n - v^*$ ;
    Remove(0); Remove( $v^*$ );
    Gen(1, 1, head,  $v^*$ , 0, TRUE);
    Add( $v^*$ ); Add(0);
  end;
  for  $i \in \{0, 1, 2, \dots, 2n - 1\}$  do  $a_i := n$ ;
  Print();
end;

```

FIG. 6. *FastChords()*.

otherwise. The routine `InitList()` initializes the list to contain every position from 0 to $2n - 1$. The function `Print()` prints out the contents of the string $\alpha = a_0a_1 \cdots a_{2n-1}$.

The various details of the functions `FastChords()`, `Gen($t, p, s, v^*, last, B$)`, `Gen2($t, p, s, v, p', part$)`, and `GenRest(s, e, v)` are described in the following subsections. Many of the details correspond directly to comments made in the verification algorithm.

6.2.1. `FastChords()`. The routine `FastChords()` drives the algorithm by calling `Gen(1, 1, head, v^* , 0, TRUE)` for each value v^* ranging from 1 to $n - 1$. Before making the call, it makes the first assignment of the value v^* to the position a_0 as well as the assignment of the value $2n - v^*$ to the position a_{v^*} . The only string with minimum value n is $\alpha = n^{2n}$. This string is listed separately at the end of this function. The pseudocode for `FastChords()` is shown in Figure 6.

6.2.2. `Gen($t, p, s, v^*, last, B$)`. This function generates all necklaces pos'_{v^*} by recursively going through each available position s in α and attempting to place the value v^* . The function maintains the following parameters (the first two are from the necklace generation algorithm):

- t : maintains the length of the prenecklace pos'_{v^*}
- p : maintains the length of the longest Lyndon prefix of pos'_{v^*}
- s : the position of α to be filled
- v^* : the value to be placed into position s
- $last$: the position of the last inserted value v^* in α
- B : boolean value indicating if it the first time the prenecklace pos'_{v^*} has been encountered

At each call to `Gen($t, p, s, v^*, last, B$)`, the string $pos'_{v^*} = q_1q_2 \cdots q_{t-1}$ is a prenecklace. To extend this string to a length t prenecklace, the next value q_t must be at least q_{t-p} . If we set the value $min = last + q_{t-p}$ then if $s \geq min$, then the new value $s - last$ can be appended to the prenecklace of length $t - 1$ (as long as the associated position $(s + v^*) \bmod 2n$ is available) to obtain a new prenecklace of length t .

Before we attempt to extend the pre-necklace pos'_{v^*} we must consider its status with the value $2n - last$ appended to the end. If $min < 2n$, then the string with the appended value is a Lyndon word and for each Lyndon word we call `Gen`

```

procedure Gen (  $t, p, s, v^*, last$ : integer;  $B$ : boolean );
local  $s', e, min$  : integer;
begin
   $min := last + pos'_{v^*, t-p}$ ;
  if  $min < 2n$  and  $B = \text{TRUE}$  then GenRest( $head, head.next, v^* + 1$ );
  if  $min = 2n$  and  $t \bmod p = 0$  and  $B = \text{TRUE}$  then begin
    if  $t = n$  then Print();
    else Gen2( $1, 1, head, v^* + 1, 2n\frac{p}{t}, 0$ );
  end;
  if  $min < 2n$  and  $s < 2n$  then begin
     $e := (s + v^*) \bmod 2n$ ;
    if  $s \geq min$  and Avail( $e$ ) then begin
       $s' := s.next$ ;
      if  $s' = e$  then  $s' := e.next$ ;
       $a_s := v^*$ ;  $a_e := 2n - v^*$ ;
      Remove( $s$ ); Remove( $e$ );
       $pos'_{v^*, t} := s - last$ ;
      if  $s = min$  then Gen( $t + 1, p, s', v^*, s, \text{TRUE}$ );
      else Gen( $t + 1, t, s', v^*, s, \text{TRUE}$ );
      Add( $e$ ); Add( $s$ );
    end;
    Gen( $t, p, s.next, v^*, last, \text{FALSE}$ );
  end;
end; end;

```

FIG. 7. $Gen(t, p, s, v^*, last, B)$.

Rest($head, head.next, v^* + 1$) to fill the remainder of the string α . If $min = 2n$ and $t \bmod p = 0$, then the modified string is a periodic necklace and for each periodic necklace we attempt to place the value $v^* + 1$ by calling Gen2($1, 1, head, v^* + 1, 2n\frac{p}{t}, 0$), unless α has been completely filled ($t = n$), in which case we simply print the string. For these tests we must be careful not to consider the same prenecklace pos'_{v^*} twice. The boolean value B indicates whether or not the prenecklace has been encountered before. If B is TRUE, then it is the first time the prenecklace has been encountered.

Once we have checked if pos'_{v^*} with the appended value $2n - last$ is a necklace, we proceed by attempting to add the value v^* to the position s . If we can, then we remove the positions s and $(s + v^*) \bmod 2n$ from the avail list, make the appropriate assignments to α and pos'_{v^*} , and make a recursive call with appropriate updates to the parameters. Finally, regardless of whether or not a value has been placed, we make a recursive call for the next available position $s.next$, but here we must set the boolean value B to FALSE, since the same prenecklace is used in the resulting recursive call.

This function assumes that the first position in the string α has been assigned the value v^* . The pseudocode for Gen($t, p, s, v^*, last, B$) is shown in Figure 7.

6.2.3. Gen2($t, p, s, v, p', part$). This function generates all necklaces pos'_v for values $v > v^*$, given the remaining available positions in the string α . It maintains the following parameters:

- t : maintains the length of the prenecklace pos'_v
- p : maintains the length of the longest Lyndon prefix of pos'_v
- s : the position of α to be filled
- v : the value to be placed into position s

- p' : the value as described in the verification algorithm
- $part$: maintains the number of times p' has been inserted

According to the verification algorithm, we must make two modifications to the string pos_v . First we convert all positions s in the string to $s \bmod p'$. Second we must insert the values p' at particular locations in the string. Thus, if we generate the prenecklaces pos_v by converting each position s to $s \bmod p'$ and adding the values p' where necessary, we are in fact generating the prenecklaces pos'_v .

The extension of the prenecklaces $pos'_v = q_1q_2 \cdots q_{t-1}$ is similar to the previous function, but in this case the value min is simply q_{t-p} and the value we wish to add is $s \bmod p'$. Thus if $s \bmod p' \geq min$ and the associated position $(s + v) \bmod 2n$ is available, then we can extend the prenecklace pos'_v in a similar fashion to $Gen(t, p, s, v^*, last, B)$.

Once we have considered all available positions s in α , the parameter s will equal $2n$. If $head = 2n$, then α is full and by construction it is in canonical form. In this case the string α is printed. Otherwise, we analyze the string pos'_v to see if it is a Lyndon word or a periodic necklace. If min is strictly less than p' , then the string is a Lyndon word and $GenRest(head, head.next, v + 1)$ is called to fill the remaining available positions in α . If $t \bmod p = 0$, then the string is a periodic necklace. In this case we call $Gen2(1, 1, head, v + 1, 2n \frac{p}{t}, 0)$ to generate the necklaces pos'_{v+1} . Before we make the initial test of $s = 2n$, however, we must consider three special cases.

Case 1. When $v = n$ we do not want to place the value v in any position s greater than n . This is because the resulting string is equivalent to placing the value in the position $n - s$. Thus as soon as we reach such a state we terminate generation from this node, unless α is full ($head = 2n$), in which case we print the string.

Case 2. We must consider the case when the value v is not placed in the string α . This state occurs when $t = 1$ and $s > p'$. In this case we continue with the placement of $v + 1$ by calling $Gen2(1, 1, head, v + 1, p', 0)$. Before making this call, we must make sure that v is less than n . Otherwise, we will end up trying to place a value greater than n which will result in the value $2n - v$, which will be less than n , being added to α .

Case 3. The final case to consider is the placement of the values p' in the string pos'_v . These values are placed the first time the position s exceeds the value $p'(part + 1)$. Once the value is added, then the generation is continued by incrementing the parameter $part$ by 1 and updating the values t and p as usual.

The pseudocode for $Gen2(t, p, s, v, p', part)$ is shown in Figure 8.

6.2.4. $GenRest(s, e, v)$. The routine $GenRest(s, e, v)$ is a simple recursive procedure that fills the remaining available positions in α with values greater than or equal to v . It takes the following parameters as input:

- s : the first available position in α
- e : another available position in α
- v : the minimum value to be placed

The idea is to place a chord joining positions s and e . Such an assignment is valid as long as the values $e - s$ and $2n - e + s$ are both greater than or equal to v . If the assignment is valid, then a recursive call is made with the next two available positions. Regardless, if the assignment is valid, we make a recursive call to check the next possible position for e which is $e.next$. If $2n - e.next + s < v$, then clearly no empty positions past e will provide valid assignments. Once all the positions are filled

```

procedure Gen2 (  $t, p, s, v, p', part$ : integer );
local  $s', e, min$ : integer;
begin
   $min := pos'_{v,t-p}$ ;
  if  $v = n$  and  $s > n$  then begin
    if  $head = 2n$  then Print();
  end;
  else if  $t = 1$  and  $s > p'$  then begin
    if  $v < n$  then Gen2(1, 1,  $head, v + 1, p', 0$ );
  end;
  else if  $s > p'(part + 1)$  then begin
     $pos'_{v,t} := p'$ ;
    if  $min = p'$  then Gen2( $t + 1, p, s, v, p', part + 1$ );
    else Gen2( $t + 1, t, s, v, p', part + 1$ );
  end;
  else if  $s = 2n$  then begin
    if  $head = 2n$  then Print();
    else if  $min < p'$  then GenRest( $head, head.next, v + 1$ );
    else if  $t \bmod p = 0$  then Gen2(1, 1,  $head, v + 1, 2n \frac{p}{t}, 0$ );
  end;
  else begin
     $e := (s + v) \bmod 2n$ ;
    if  $s \bmod p' \geq min$  and Avail( $e$ ) then begin
       $s' := s.next$ ;
      if  $s' = e$  then  $s' := e.next$ ;
       $a_s := v$ ;  $a_e := 2n - v$ ;
      Remove( $s$ ); Remove( $e$ );
       $pos'_{v,t} := s \bmod p'$ ;
      if  $s \bmod p' = min$  then Gen2( $t + 1, p, s', v, p', part$ );
      else Gen2( $t + 1, t, s', v, p', part$ );
      Add( $e$ ); Add( $s$ );
    end;
    Gen2( $t, p, s.next, v, p', part$ );
  end; end;

```

FIG. 8. $Gen2(t, p, s, v, p', part)$.

($s = 2n$) then the string is printed. The pseudocode for $GenRest(s, e, v)$ is shown in Figure 9.

6.2.5. Analysis. As with the previous algorithm, we obtain experimental results for the amount of work done compared to the number of chord diagrams generated. Since the work done for each recursive call is constant, we count the amount of work done by summing the number of recursive calls. The resulting ratios are shown in Table 3 for $n \leq 12$. Notice that the ratios are decreasing (after $n = 5$) as the number of chords increases. This gives a very strong indication that the algorithm runs in constant amortized time.

```

procedure GenRest ( s, e, v : integer );
begin
  if s = 2n then Print();
  else if e ≠ 2n then begin
    if e - s ≥ v and 2n - e + s ≥ v then begin
      as := e - s;   ae := 2n - as;
      Remove(s); Remove(e);
      GenRest(head,head.next,v);
      Add(e); Add(s);
    end;
    if 2n - e.next + s ≥ v then GenRest(s,e.next,v);
  end; end;

```

FIG. 9. *GenRest*(*s, e, v*).

TABLE 3
Experimental results for FastChords().

Number of chords <i>n</i>	Nonisomorphic chord diagrams	Ratio of work done to chord diagrams generated
1	1	1.0
2	2	3.0
3	5	9.2
4	18	13.6
5	105	14.2
6	902	12.4
7	9749	11.0
8	127072	10.0
9	1915951	9.4
10	32743182	8.9
11	624999093	8.5
12	13176573910	8.2

CONJECTURE 1. *The algorithm for generating nonisomorphic chord diagrams, FastChords(), is CAT.*

A complete C program for each of the nonisomorphic chord diagram generation algorithms is available from the author. Table 4 shows the outputs from each of the two algorithms for values of *n* up to 4.

7. Future work. In this paper we have outlined a fast algorithm for generating nonisomorphic chord diagrams. However, we have not found a mathematical proof to show that the algorithm is CAT, leaving a challenging open problem. We have also mentioned two other open problems in this paper:

- the development of an efficient algorithm to generate *k*-ary unlabeled necklaces;
- the development of an efficient algorithm to generate *k*-ary necklaces where the number of occurrences of each alphabet symbol is fixed.

The canonical form used in the algorithm *FastChords()* has recently been used to develop a CAT algorithm for the latter problem if the number of occurrences of some value *v* is relatively prime to *n* [10].

TABLE 4
Different outputs for the two generation algorithms.

Output from SimpleChords(t, p)	Output from FastChords()
$n = 1:$ 1 1	$n = 1:$ 1 1
$n = 2:$ 1 3 1 3 2 2 2 2	$n = 2:$ 1 3 1 3 2 2 2 2
$n = 3:$ 1 5 1 5 1 5 1 5 2 2 4 4 1 5 3 1 5 3 2 3 4 2 3 4 3 3 3 3 3 3	$n = 3:$ 1 5 1 5 1 5 1 5 3 1 5 3 1 5 2 2 4 4 2 3 4 2 3 4 3 3 3 3 3 3
$n = 4:$ 1 7 1 7 1 7 1 7 1 7 1 7 2 2 6 6 1 7 1 7 3 1 7 5 1 7 2 3 6 2 5 6 1 7 2 4 6 1 7 4 1 7 3 3 3 5 5 5 1 7 3 4 2 5 6 4 1 7 4 2 3 6 4 5 1 7 4 4 1 7 4 4 1 7 5 2 2 6 6 3 1 7 5 3 1 7 5 3 2 2 6 6 2 2 6 6 2 3 6 3 5 2 5 6 2 4 6 3 3 4 5 5 2 4 6 4 2 4 6 4 3 4 4 5 3 4 4 5 3 5 3 5 3 5 3 5 4 4 4 4 4 4 4 4	$n = 4:$ 1 7 1 7 1 7 1 7 1 7 1 7 3 1 7 5 1 7 1 7 2 2 6 6 1 7 4 1 7 2 4 6 1 7 5 3 1 7 5 3 1 7 4 4 1 7 4 4 1 7 2 3 6 2 5 6 1 7 3 3 3 5 5 5 1 7 3 4 2 5 6 4 1 7 4 2 3 6 4 5 1 7 4 4 1 7 4 4 1 7 4 2 3 6 4 5 1 7 5 2 2 6 6 3 1 7 5 3 1 7 5 3 2 2 6 6 2 2 6 6 2 5 6 2 3 6 3 5 2 4 6 4 2 4 6 4 2 4 6 3 3 4 5 5 3 5 3 5 3 5 3 5 3 4 4 5 3 4 4 5 4 4 4 4 4 4 4 4

REFERENCES

- [1] D. BAR-NATAN, *On the Vassiliev knot invariants*, *Topology*, 34 (1995), pp. 423–472.
- [2] J. BIRMAN AND R. TRAPP, *Braided chord diagrams*, *J. Knot Theory Ramifications*, 7 (1998), pp. 1–22.
- [3] K. CATTELL, F. RUSKEY, J. SAWADA, C.R. MIERS, AND M. SERRA, *Fast algorithms to generate necklaces, unlabeled necklaces, and irreducible polynomials over GF(2)*, *J. Algorithms*, 37 (2000), pp. 267–282.
- [4] R. CORI AND M. MARCUS, *Counting non-isomorphic chord diagrams*, *Theoret. Comput. Sci.*, 204 (1998), pp. 55–73.
- [5] J-P. DUVAL, *Factoring words over an ordered alphabet*, *J. Algorithms*, 4 (1983), pp. 363–381.
- [6] J. HARER AND D. ZAGIER, *The Euler characteristic of the moduli space of curves*, *Invent. Math.*, 85 (1986), pp. 457–485.
- [7] J.E. KOEHLER, *Folding a strip of stamps*, *J. Combinatorial Theory*, 5 (1968), pp. 135–152.
- [8] B. LI AND H. SUN, *Exact number of chord diagrams and an estimation of the number of spine diagrams of order n* , *Chinese Sci. Bull.*, 42 (1997), pp. 705–720.
- [9] F. RUSKEY AND J. SAWADA, *An efficient algorithm for generating necklaces of fixed density*, *SIAM J. Comput.*, 29 (1999), pp. 671–684.
- [10] J. SAWADA, *Fast Algorithms to Generate Restricted Classes of Strings under Rotation*, Dissertation, University of Victoria, Victoria, BC, Canada, 2000.
- [11] J. SAWADA, *Generating bracelets in constant amortized time*, *SIAM J. Comput.*, 31 (2001), pp. 259–268.

- [12] A. STOIMENOW, *Enumeration of chord diagrams and an upper bound for Vassiliev invariants*, J. Knot Theory Ramifications, 7 (1998), pp. 93–114.
- [13] A. STOIMENOW, *On the number of chord diagrams*, Discrete Math., 218 (2000), pp. 209–233.
- [14] T. WALSH AND A. LEHMAN, *Counting rooted maps by genus I*, J. Combinatorial Theory Ser. B, 13 (1972), pp. 192–218.
- [15] T. WALSH AND A. LEHMAN, *Counting rooted maps by genus II*, J. Combinatorial Theory Ser. B, 13 (1972), pp. 122–141.

INVERTING RANDOM FUNCTIONS II: EXPLICIT BOUNDS FOR DISCRETE MAXIMUM LIKELIHOOD ESTIMATION, WITH APPLICATIONS*

MICHAEL A. STEEL[†] AND LÁSZLÓ A. SZÉKELY[‡]

Abstract. In this paper we study inverting random functions under the maximum likelihood estimation (MLE) criterion in the discrete setting. In particular, we consider how many independent evaluations of the random function at a particular element of the domain are needed for reliable reconstruction of that element. We provide explicit upper and lower bounds for MLE, both in the nonparametric and parametric setting, and give applications to coin-tossing and phylogenetic tree reconstruction.

Key words. random function, maximum likelihood estimation, Kullback–Leibler distance, phylogeny reconstruction

AMS subject classifications. 62B10, 62C05, 62F10, 92D15

PII. S089548010138790X

1. Review of random functions. This paper is a sequel of our earlier paper [12]. We assume that the reader is familiar with that paper; however, we repeat the most important definitions.

For two finite sets, A and U , let us be given a U -valued random variable ξ_a for every $a \in A$. We call the vector of random variables $(\xi_a : a \in A)$ a *random function* $\Xi : A \rightarrow U$. Ordinary functions are specific instances of random functions. It is easy to see [12] that an equivalent definition of random functions is obtained by picking one of the $|U|^{|A|}$ ordinary functions from A to U according to some distribution.

Given another random function, Γ , from U to V , we can speak about the composition of Γ and Ξ , $\Gamma \circ \Xi : A \rightarrow V$, which is the vector variable $(\gamma_{\xi_a} : a \in A)$. In this paper we are concerned with inverting random functions. In other words, we look for random functions $\Gamma : U \rightarrow A$ in order to obtain the best approximations of the identity function $\iota : A \rightarrow A$ by $\Gamma \circ \Xi$. *We always assume that Ξ and Γ are independent.* This assumption holds for free if either Ξ or Γ is a deterministic function.

Our motivation for the study of random functions came from phylogeny reconstruction. Stochastic models define how biomolecular sequences are generated at the leaves of a binary tree. If all possible binary trees on n leaves come equipped with a model for generating biomolecular sequences of length k , then we have a random function from the set of binary trees with n leaves to the ordered n -tuples of biomolecular sequences of length k . *Phylogeny reconstruction* is a random function from the set of ordered n -tuples of biomolecular sequences of length k to the set of binary trees with n leaves. It is a natural assumption that random mutations in the past are independent from any random choices in the phylogeny reconstruction algorithm. Criteria for phylogeny reconstruction may differ according to what one wishes to optimize.

*Received by the editors April 12, 2001; accepted for publication (in revised form) June 16, 2002; published electronically September 10, 2002.

<http://www.siam.org/journals/sidma/15-4/38790.html>

[†]Biomathematics Research Centre, University of Canterbury, Christchurch, New Zealand (m.steel@math.canterbury.ac.nz). This author was supported by the New Zealand Marsden Fund.

[‡]Department of Mathematics, University of South Carolina, Columbia, SC (szekely@math.sc.edu). This author was supported by NSF grants DMS 9701211 and 0072187, by a grant of SC CHE, and Hungarian NSF grant T 032455. This research was started when this author visited the University of Canterbury under the support of the New Zealand Marsden Fund.

Consider the probability of returning a from a by the composition of two random functions; that is, $r_a = \mathbb{P}[\gamma_{\xi_a} = a]$. The assumption on the independence of Ξ and Γ immediately implies

$$(1.1) \quad r_a = \sum_{u \in U} \mathbb{P}[\xi_a = u] \cdot \mathbb{P}[\gamma_u = a].$$

A natural criterion is to find Γ for a given Ξ in order to maximize $\sum_a r_a$. More generally, we may have a weight function $w : A \rightarrow \mathbb{R}^+$, and we may wish to maximize $\sum_a r_a w(a)$. This can happen if we give preference to returning certain a 's, or if we have a prior probability distribution on A and we want to maximize the expected return probability for a random element of A selected according to the prior distribution. A random function $\Gamma^* : U \rightarrow A$ can be defined in the following way: for any fixed $u \in U$,

$$(1.2) \quad \gamma_u^* = a^* \text{ for sure if } \forall a \in A, \mathbb{P}[\xi_{a^*} = u]w(a^*) \geq \mathbb{P}[\xi_a = u]w(a).$$

In case there is more than one element a^* that satisfies (1.2), we may select uniformly at random from the set of such elements. This function Γ^* is called the *maximum a posteriori estimator* (MAP) in the literature [7]. The special case when the weight function w is constant is known as the *maximum likelihood estimation* (MLE) [2, 7]. The MAP estimator Γ^* maximizes $\sum_a r_a w(a)$ for any given Ξ ; i.e., MAP is best on average. This result appears as Theorem 17.2 of [8], but an equivalent formulation, in the context of decision theory, is given by Theorem 10.3.1 of [2]; a further formulation, using a different proof, appears as Theorem 3.1 of [12]. However, it is at least as natural to look at a more conservative criterion: maximize the *smallest* value of r_a for $a \in A$; i.e., do the worst case the best. For this criterion MAP or MLE is, in general, not optimal. It is surprising, but little is known about the performance of MAP or MLE under this more conservative criterion.

Our paper [12] introduced a new abstract model for phylogeny reconstruction: inverting parametric random functions. Most of the work done on the mathematics of phylogeny reconstruction can be discussed in this context. This model is more structured than random functions, and hence is better suited to describe details of models of phylogeny and the evolution of biomolecular sequences. The approach is likely to be applicable in other areas where “nuisance” parameters are involved.

Assume that for a finite set A , for every $a \in A$, an (arbitrary, finite, or infinite) set $\Theta(a) \neq \emptyset$ is assigned, and, moreover, $\Theta(a) \cap \Theta(b) = \emptyset$ for $a \neq b$. Set $B = \{(a, \theta) : a \in A, \theta \in \Theta(a)\}$ and let π_1 denote the natural projection from B to A . A *parametric random function* is the collection Ξ of random variables such that

(i) for $a \in A$ and $\theta \in \Theta(a)$ there is a (unique) U -valued random variable $\xi_{(a,\theta)}$ in Ξ .

We are interested in random functions $\Gamma : U \rightarrow A$ independent from Ξ so that $\gamma_{\xi_{(a,\theta)}}$ best approximates π_1 under certain criteria. Call $R_{(a,\theta)}$ the probability $\mathbb{P}[\gamma_{\xi_{(a,\theta)}} = a]$. MLE, as it is used in the practice of phylogeny reconstruction, would take the Γ' , for which, for every fixed u , $\gamma'_u = a'$ for sure, if

$$(1.3) \quad \forall (a, \theta) \in B \quad \exists \theta' \in \Theta(a') \quad \mathbb{P}[\xi_{(a',\theta')} = u] \geq \mathbb{P}[\xi_{(a,\theta)} = u].$$

In case there is more than one element a' that satisfies (1.3), we may select uniformly at random from the set of such elements. (We avoided using the more natural looking quantification $\exists \theta' \in \Theta(a')$ for all $(a, \theta) \in B$, since $\mathbb{P}[\xi_{(a',\theta')} = u]$ may not take a

maximum value!) We denote by $R'_{(a,\theta)}$ the probability that from the pair (a, θ) the MLE Γ' returns a , i.e.,

$$(1.4) \quad R'_{(a,\theta)} = \mathbb{P}[\gamma'_{\xi_{(a,\theta)}} = a].$$

In [12] we made further assumptions on parametric random functions that we do not make in this paper:

(ii) There is a measure space $(\Theta(a), \mu_a(\cdot))$ defined on every $\Theta(a)$ such that $\mu_a(\Theta(a)) < \infty$.

(iii) For all $u \in U$, and for all $a \in A$, $\mathbb{P}[\xi_{(a,\theta)} = u] \in L^1(\Theta(a), \mu_a(\cdot))$.

Under these additional conditions we showed in [12] that in the model of parametric random functions the MLE criterion has to be modified to ensure the property that Γ' maximizes:

$$(1.5) \quad \sum_{a \in A} \int R_{(a,\theta)} d\mu_a(\theta).$$

This criterion is natural, since if $\sum_{a \in A} \int d\mu_a(\theta) = 1$, the formula (1.5) can be interpreted as the expected probability of return of elements of A , given a prior distribution on A .

The purpose of this paper is to place explicit upper and lower bounds on the probability that MLE correctly reconstructs elements of A , in both the parametric and nonparametric settings. Our primary interest is in the situation where k independent experiments are carried out, and we wish to determine how large k needs to be in order to correctly recover the underlying element of A with high probability. To emphasize the role of k we will let $[r^{(k)}]_a^*$ (resp., $[R^{(k)}]_{(a,\theta)}'$) denote the probability that MLE correctly reconstructs a in the nonparametric (resp., parametric) setting. We illustrate our bounds in the nonparametric setting by applications to coin-tossing and phylogeny reconstruction.

For the parametric setting, we first show, by way of an example, that the nonparametric upper bound on k does not extend in the way one might hope or expect. Nevertheless, we provide (in Theorem 5.1) an explicit upper bound on the number k of experiments required for MLE to reconstruct elements of A accurately. This result can be regarded as an extension of a discrete version of Wald's theorem [15]. We describe some implications of this result for phylogeny reconstruction in the remarks following Theorem 5.1.

Most of present paper can be considered as an attempt to analyze the worst case behavior of MLE. This is a very natural question in situations where a prior distribution is not given on A , or the inverting of the random function is to be carried out only once. Such a situation arises in phylogeny reconstruction, where we do not have a prior distribution on alternative evolutionary scenarios, and the reconstruction is not going to be repeated—there is only one “tree of life” that we want to know. However, the results in this paper are not restricted to the phylogeny setting and may be relevant to several other areas where MLE estimation is employed.

Our approach is information-theoretic; we focus on the possibility or impossibility of inverting random functions, and not on the computational complexity issues. Our results can also be restated in the language of decision theory, by talking about “loss functions” and “risk function” associated with the ML decision rule. Although some of our consequences or applications (described in section 4) may be derivable from existing theory, as far as we are aware the main results in this paper are not special cases of published results in either information theory or statistical decision theory.

2. Distances between distributions. For $a, b \in A, \Xi : A \rightarrow U$, let

$$(2.1) \quad d(a, b) = \sum_{u \in U} |\mathbb{P}[\xi_a = u] - \mathbb{P}[\xi_b = u]|.$$

We will refer to $d(a, b)$ as the *variational distance* of the random variables ξ_a and ξ_b . We also use the *Hellinger distance* of the random variables ξ_a and ξ_b , defined by

$$(2.2) \quad d_H(a, b) = \sqrt{\sum_{u \in U} \left(\sqrt{\mathbb{P}[\xi_a = u]} - \sqrt{\mathbb{P}[\xi_b = u]} \right)^2}.$$

These measures sometimes appear with slightly different definitions, terminology, and normalization constants. (For example, $\frac{1}{2}d(a, b)$ is sometimes referred to as the “variation distance.”) It is well known (see p. 25 in [9]) that $0 \leq d(a, b) \leq 2$ and

$$(2.3) \quad d_H^2(a, b) \leq d(a, b) \leq 2d_H(a, b).$$

We are going to use a well known and elegant multiplicative property of the Hellinger distance. For any $\Xi : A \rightarrow U$ random function define the $\Xi^{(k)} : A \rightarrow U^k$ random function as a sequence of k independent trials of Ξ . Let $d_H^{(k)}(a, b)$ denote the Hellinger distance of the random variables $\xi_a^{(k)}$ and $\xi_b^{(k)}$. Then independence immediately implies the identity

$$(2.4) \quad 1 - \frac{1}{2} \left(d_H^{(k)}(a, b) \right)^2 = \left(1 - \frac{1}{2} d_H^2(a, b) \right)^k,$$

by virtue of the formula

$$(2.5) \quad \sum_{u \in U} \left(\sqrt{\mathbb{P}[\xi_a = u]} - \sqrt{\mathbb{P}[\xi_b = u]} \right)^2 = 2 - 2 \sum_{u \in U} \sqrt{\mathbb{P}[\xi_a = u]} \sqrt{\mathbb{P}[\xi_b = u]}.$$

Combining the inequality $1 - (1 - x)^k \leq kx$ which holds for all $0 \leq x \leq 1$ and k positive integers, and (2.4), we obtain

$$(2.6) \quad \left(d_H^{(k)}(a, b) \right)^2 = 2 \left[1 - \left(1 - \frac{1}{2} d_H^2(a, b) \right)^k \right] \leq k d_H^2(a, b).$$

Using the notation $d^{(k)}(a, b)$ for the variational distance of the k independent trials, i.e., of the random variables $\xi_a^{(k)}$ and $\xi_b^{(k)}$, inequalities (2.3) and (2.6) imply

$$(2.7) \quad d^{(k)}(a, b) \leq 2\sqrt{k}d_H(a, b).$$

The nonsymmetric Kullback–Leibler distance (or relative entropy) of the random variables ξ_a and ξ_b is defined as

$$d_{KL}(a, b) = \sum_{u \in U} \mathbb{P}[\xi_a = u] \log \frac{\mathbb{P}[\xi_a = u]}{\mathbb{P}[\xi_b = u]}.$$

We will use the inequality [4]

$$(2.8) \quad d_{KL}(a, b) \geq \frac{1}{2}d^2(a, b).$$

3. MLE for inverting random functions. In this section we describe some lower and upper bounds on the probability that MLE correctly reconstructs elements of the set A . A classical upper bound on the average value of r_a over A —or more generally the value of $\sum_{a \in A} r_a w(a)$ for some probability distribution w on A —is given by “Fano’s inequality” (see, for example, [4]). Here we recall from [12] a different type of upper bound that applies also to r_a for any particular value of a and which is closely related to the variational distance.

THEOREM 3.1. *Assume that we have finite sets A and U and random functions $\Xi : A \rightarrow U$ and $\Gamma : U \rightarrow A$. Suppose that there is an element $b \in A$ and a subset N , $b \in N \subset A$ such that for all $a \in N$*

$$d(a, b) < \delta.$$

Then we have

$$\min_{a \in N} r_a \leq \frac{1}{|N|} + \delta \left(1 - \frac{1}{|N|} \right).$$

Now we can state the following lower bound for r_a in the setting of Theorem 3.1.

THEOREM 3.2. *Assume that we have finite sets A and U and a random function $\Xi : A \rightarrow U$. Assume that $\Gamma^* : U \rightarrow A$ is the MLE, and r_a^* is the return probability of $a \in A$ using Γ^* . Then we have*

$$(3.1) \quad r_a^* \geq 1 - \sum_{b \neq a} \left(1 - \frac{1}{2} d(a, b) \right).$$

If the MLE $\Gamma^* : U^k \rightarrow A$ is applied to invert the random function $\Xi^{(k)} : A \rightarrow U^k$, which is a sequence of k independent trials of Ξ , then

$$(3.2) \quad [r^{(k)}]_a^* \geq 1 - \sum_{b \neq a} \left(1 - \frac{1}{2} d_H^2(a, b) \right)^k.$$

Proof. For $y \in A$ let

$$U_y = \{u \in U \mid \forall x \in A, x \neq y, \mathbb{P}[\xi_y = u] > \mathbb{P}[\xi_x = u]\}$$

and similarly for V_y with “ \geq ” instead of “ $>$ ” in the definition. It is clear from independence (1.1) and the definition (1.2) that

$$(3.3) \quad r_a^* \geq \sum_{u \in U_a} \mathbb{P}[\xi_a = u].$$

For $x, y \in A$ set $p_y^x = \sum_{u \in V_y} \mathbb{P}[\xi_x = u]$. Now we claim

$$(3.4) \quad r_a^* \geq 1 - \sum_{y \neq a} p_y^a.$$

Note that

$$\sum_{y \neq a} p_y^a = \sum_{y \neq a} \sum_{u \in V_y} \mathbb{P}[\xi_a = u] \geq \mathbb{P}[\xi_a \notin U_a],$$

since the complement of U_a is a subset of $\cup_{y \neq a} V_a$, and

$$\mathbb{P}[\xi_a \notin U_a] = 1 - \mathbb{P}[\xi_a \in U_a] \geq 1 - r_a^*$$

by (3.3). This establishes (3.4). Finally, we have

$$\begin{aligned} d(a, y) &= \sum_{u \in U} |\mathbb{P}[\xi_a = u] - \mathbb{P}[\xi_y = u]| \\ &= \sum_{u \in V_y} (\mathbb{P}[\xi_y = u] - \mathbb{P}[\xi_a = u]) + \sum_{u \notin V_y} |\mathbb{P}[\xi_a = u] - \mathbb{P}[\xi_y = u]| \\ &\leq p_y^y - p_y^a + \sum_{u \notin V_y} (\mathbb{P}[\xi_a = u] + \mathbb{P}[\xi_y = u]) = p_y^y - p_y^a + (1 - p_y^a) + (1 - p_y^y) = 2 - 2p_y^a. \end{aligned}$$

Hence, $p_y^a \leq 1 - \frac{1}{2}d(a, y)$, and plugging this into (3.4) yields (3.1). To prove (3.2), apply (3.1) to $\Xi^{(k)}$ and invoke (2.4). \square

Remarks. First, note that (3.2) immediately implies that if $d_a = \min_{b \neq a} d_H(a, b)$, then $[r^{(k)}]_a^* > 1 - |A| \exp(-kd_a^2/2)$. Consequently, if

$$(3.5) \quad k > \frac{2}{d_a^2} \log \frac{|A|}{\epsilon},$$

then $[r^{(k)}]_a^* > 1 - \epsilon$. Second, note that an analogue of (3.2) also holds if, instead of k independent trials of Ξ , we take independent $A \rightarrow U$ random functions $\Xi_1, \Xi_2, \dots, \Xi_k$. Now the lower bound on $[r^{(k)}]_a^*$ is

$$1 - \sum_{b \neq a} \prod_{i=1}^k \left(1 - \frac{1}{2} d_H^2((\xi_i)_a, (\xi_i)_b)\right).$$

4. Applications.

4.1. Solving biased coin-tossing with MLE. We want to show an example where our upper and lower bounds for reconstructing random functions are nearly tight. Assume that $U = \{T, H\}$; i.e., we are tossing coins. Let a set A consist of $n + 1$ biased coins, denoted by $0, 1, 2, \dots, n$. Define the random function Ξ as follows: coin i shows H with probability i/n and shows T with probability $1 - i/n$. We show the following: there is a constant c_1 such that for $k = c_1 n^2$, for k independent trials of Ξ , $\Xi^{(k)}$, $[r^{(k)}]_i$ cannot be uniformly close to 1, no matter which method is used for inverting $\Xi^{(k)}$. However, there is a constant c_2 such that for $k = c_2 n^2$, using MLE, we find $[r^{(k)}]_i^*$ uniformly close to 1.

For simplicity we assume that n is odd. We are going to use Theorem 3.1 in the following setting: $b = \frac{n-1}{2}$, $N = \{\frac{n-3}{2}, \frac{n-1}{2}, \frac{n+1}{2}\}$. Then,

$$\min_{a \in N} [r^{(k)}]_a \leq \frac{1}{3} + \frac{2}{3}\delta,$$

where δ is the largest variational distance for $\Xi^{(k)}$ among b and the elements of N . Observe that for Ξ , by formula (2.5), we have

$$(4.1) \quad d_H^2(i, j) = 2 \left(1 - \frac{\sqrt{ij}}{n} - \frac{\sqrt{(n-i)(n-j)}}{n}\right).$$

It is easy to see that, for $i = b, j \in N$, (4.1) is maximized by $j_0 = \frac{n+1}{2}$ at the value $2(1 - \sqrt{1 - \frac{1}{n^2}}) \leq 2/n^2$. By (2.7), $d^{(k)}(b, x) \leq 2\sqrt{k}d_H(b, x)$, for every x , and therefore $\delta \leq 2\sqrt{k}d_H(b, j_0) \leq 4\sqrt{k}/n$. Any choice of $c_1 < 1/4$ suffices to keep either $r_{\frac{n-3}{2}}$ or $r_{\frac{n+1}{2}}$ separated from 1.

In the other direction we use Theorem 3.2. By (3.2) and (4.1) we have

$$(4.2) \quad [r^{(k)}]_i^* \geq 1 - \sum_{\substack{j=0 \\ j \neq i}}^n \left(1 - \frac{1}{2}d_H^2(i, j)\right)^k = 1 - \sum_{\substack{j=0 \\ j \neq i}}^n \left(\frac{\sqrt{ij}}{n} + \frac{\sqrt{(n-i)(n-j)}}{n}\right)^k.$$

By the classical inequality

$$\sqrt{a_1 a_2} + \sqrt{b_1 b_2} \leq \sqrt{a_1 + b_1} \cdot \sqrt{a_2 + b_2},$$

the generic subtracted term in the summation (4.2) is estimated from above by

$$\left(1 - \frac{(j-i)^2}{n^2}\right)^{k/2}.$$

Hence,

$$(4.3) \quad [r^{(k)}]_i^* \geq 1 - 2 \sum_{m=1}^n \left(1 - \frac{m^2}{n^2}\right)^{k/2}.$$

Now observe that

$$(4.4) \quad \sum_{m=1}^n \left(1 - \frac{m^2}{n^2}\right)^{k/2} \leq \left(1 - \frac{1}{n^2}\right)^{k/2} + n \int_{1/n}^1 (1 - x^2)^{k/2} dx$$

and

$$n \int_{1/n}^1 (1 - x^2)^{k/2} dx \leq n \int_{1/n}^1 e^{-k \cdot x^2/2} dx \leq \frac{n}{\sqrt{k}} \int_{\sqrt{k}/n}^{\sqrt{k}} e^{-t^2/2} dt \leq \frac{n}{\sqrt{k}} \cdot \frac{\sqrt{2\pi}}{2}.$$

Therefore, for a sufficiently large c_2 , selecting $k = c_2 n^2$, both terms in the right-hand side of (4.4) will be as small as wanted, and hence in (4.3) $[r^{(k)}]_i^*$ will be as close to 1 as wanted.

4.2. Phylogeny reconstruction. As a second application, we consider a problem arising in phylogenetic analysis. In this setting we have a model for generating sequences at the leaves of a tree, and the question is how long such sequences need to be in order to correctly reconstruct the tree—with high probability—from just the generated sequences.

The simplest stochastic model, for two-state sequences, is the symmetric model, due to Neyman [10], which we call the Neyman-2 model. (Related models also arise in statistical physics and in the theory of noisy communication—see, for example, [6].) Let $\{0, 1\}$ denote the two states. Let us be given a binary tree T (a tree in which each vertex has degree 1 or 3) with n labeled leaves. We describe how a single site in the sequence develops on T , and then we assume that the sites are independently and identically distributed (i.i.d.).

For each edge e of T we have an associated *transition probability*, which lies strictly between 0 and 0.5. Let $p : E(T) \rightarrow (0, 0.5)$ denote the associated map. Select one of

the leaves¹ and assign it state 0 or state 1 with probability 0.5. Direct all edges away from this leaf and recursively assign random states to the vertices of T as follows: if $e = \{u, v\}$ is directed from u to v , and u (but not v) has a state assignment, then v is assigned the same state as u with probability $1 - p_e$ or the other state with probability p_e . (In this latter case, we say there is a *transition on e* .) It is assumed that all assignments are made independently, and so the pair (T, p) determines the joint probability of any assignment of states to the vertices of T and thereby the marginal probability of any assignment of states to the leaves of T . If we independently generate k such assignments of states to the leaves of T , we obtain n sequences of length k . For this model, upper bounds on the sequence length k required to reconstruct the underlying tree were given in [5, 12]. These papers showed that, for accurate tree reconstruction, k needs to grow only quadratically in $1/f$, where f is the smallest transition probability in the tree, when other parameters are fixed. We now show that this rate of growth is not only sufficient but is also necessary.

Consider binary trees having four labeled leaves and two unlabeled interior vertices. There are three such trees (up to equivalence), and we will denote them as a, b, c . Each tree has four leaf edges (an edge incident to a leaf) and one interior edge. Take $A = \{a, b, c\}$, and let U be the set of binary functions defined on the four leaves. Assume that a, b, c are Neyman-2 trees with transition probability f on the interior edge. (We do not care what the other transition probabilities are.) Let Ξ denote the state assignment of the leaves of a, b, c under the Neyman-2 model.

THEOREM 4.1. *For the three Neyman-2 binary trees a, b, c on four leaves (as described above), and state assignment Ξ , under any method for inverting random function Ξ from k independent trials (i.e., from binary sequences of length k associated with the leaves) with success probability near 1 for all three trees, $k = \Omega(\frac{1}{f^2})$.*

Proof. We are going to prove that for f sufficiently close to 0, for some constant $C > 0$,

$$(4.5) \quad d_H(a, b) \leq Cf.$$

Now (2.7) and (4.5) imply $d^{(k)}(a, b) \leq 2Cf\sqrt{k}$, and one similarly obtains $d^{(k)}(c, b) \leq 2Cf\sqrt{k}$. So if we apply Theorem 3.1 with $N = \{a, b, c\}$,

$$(4.6) \quad \min\{r_a, r_c\} \leq \frac{1}{3} + \frac{4}{3}Cf\sqrt{k},$$

and the right-hand side of (4.6) is well separated from 1 as k is a small constant over f^2 .

To complete the proof, we have to verify (4.5). Assume that a is the tree in which the interior edge separates leaves 1, 2 from leaves 3, 4; and b is the tree in which the interior edge separates leaves 1, 3 from leaves 2, 4. By (2.2)

$$(4.7) \quad d_H(a, b)^2 = \sum_{u \in U} \left(\sqrt{\mathbb{P}[\xi_a = u]} - \sqrt{\mathbb{P}[\xi_b = u]} \right)^2,$$

where the summation goes for 16 terms which correspond to the 16 elements of U : functions with domain $\{1, 2, 3, 4\}$ and range $\{0, 1\}$. We are going to condition on the

¹One assumes that mutations of an ancestral sequence happen in this way. However, it can be shown that the selection of the special leaf has no effect on the reconstruction in the model considered here.

event Φ , denoting that there is transition on the interior edge of the tree and also for the complement of this event. For $x = a, b$ define

$$\begin{aligned} A(x, u) &= \mathbb{P}[\xi_x = u \mid \neg\Phi], \\ B(x, u) &= \mathbb{P}[\xi_x = u \mid \Phi] - \mathbb{P}[\xi_x = u \mid \neg\Phi]. \end{aligned}$$

Notice that $A(x, u)$ and $B(x, u)$ are constants that do not depend on f . Also, observe that

$$\begin{aligned} \mathbb{P}[\xi_x = u] &= \mathbb{P}[\xi_x = u \mid \neg\Phi] \cdot (1 - f) + \mathbb{P}[\xi_x = u \mid \Phi] \cdot f \\ &= \mathbb{P}[\xi_x = u \mid \neg\Phi] + f \cdot (\mathbb{P}[\xi_x = u \mid \Phi] - \mathbb{P}[\xi_x = u \mid \neg\Phi]) \\ &= A(x, u) + fB(x, u). \end{aligned}$$

It easily follows from the geometry of the trees a and b that $A(a, u) = A(b, u)$. Furthermore, it is easily seen that $A(a, u) \neq 0$ for all values of u , which ensures (below) that we may divide expressions by $A(a, u)$. Hence, by the Taylor expansion of the square root function, we have

$$\begin{aligned} \sqrt{\mathbb{P}[\xi_a = u]} - \sqrt{\mathbb{P}[\xi_b = u]} &= \sqrt{A(a, u)} \left(\sqrt{1 + \frac{fB(a, u)}{A(a, u)}} - \sqrt{1 + \frac{fB(b, u)}{A(a, u)}} \right) \\ (4.8) \qquad \qquad \qquad &= f \frac{B(a, u) - B(b, u)}{2\sqrt{A(a, u)}} + O(f^2), \end{aligned}$$

and summing up 16 terms like (4.8) we obtain

$$d_H^2(a, b) = f^2 \sum_{u \in U} \frac{(B(a, u) - B(b, u))^2}{4A(a, u)} + O(f^3),$$

and this proves (4.5) for all

$$C > \sqrt{\sum_{u \in U} \frac{(B(a, u) - B(b, u))^2}{4A(a, u)}}. \quad \square$$

5. MLE for inverting parametric random functions. We start with an example showing that for parametric MLE there is no counterpart of (3.2); that is, there is no function $f = f(\delta, k)$ such that, for all $\delta > 0$, $\lim_{k \rightarrow \infty} f(\delta, k) = 0$ and

$$(5.1) \qquad [R^{(k)}]_{(a, \theta)}' \geq 1 - \sum_{b \neq a} f(\delta((a, \theta), b), k),$$

where

$$\delta((a, \theta), b) = \inf_{\theta' \in \Theta(b)} d_H((a, \theta), (b, \theta')).$$

Take $A = \{a_1, a_2\}$, $U = \{u_1, u_2, \dots, u_{2k^2}\}$, $\Theta(a_1) = \Theta(a_2) = U^k$. We denote a generic element of U^k by \mathbf{u} , and $\text{supp}(\mathbf{u})$ denotes the set of elements of U which occur as coordinates in \mathbf{u} . Let $B = (\{a_1\} \times \Theta(a_1)) \cup (\{a_2\} \times \Theta(a_2))$. Define the parametric random function $\Xi : B \rightarrow U$ as follows. Set $\mathbb{P}[\xi_{(a_1, \mathbf{u})} = v] = 1/|U|$ for each $v \in U$. For

$\mathbf{u} \in U^k$ and $v \in U$, set $\mathbb{P}[\xi_{(a_2, \mathbf{u})} = v] = i/k$ if v occurs at $i = i(v)$ coordinates in \mathbf{u} . Now for any $\mathbf{w}, \mathbf{u} \in U^k$ we have

$$(5.2) \quad d((a_1, \mathbf{w}), (a_2, \mathbf{u})) \geq 2 - \frac{1}{k}$$

by the calculation

$$\sum_{v \in \text{supp}(\mathbf{u})} \left(\frac{i(v)}{k} - \frac{1}{|U|} \right) + \sum_{v \notin \text{supp}(\mathbf{u})} \frac{1}{|U|} = 2 - 2 \sum_{v \in \text{supp}(\mathbf{u})} \frac{1}{|U|} \geq 2 - \frac{2k}{|U|} = 2 - \frac{1}{k}.$$

Now consider k independent trials of $\Xi, \Xi^{(k)}$. We study inverting $\Xi^{(k)}$ with parametric MLE. Note that, for any $\mathbf{u} \in U^k$,

$$\mathbb{P}[\xi_{(a_2, \mathbf{u})} = \mathbf{u}] = \prod_{i=1}^k \mathbb{P}[\xi_{(a_2, \mathbf{u})} = u_i] \geq \left(\frac{1}{k} \right)^k ;$$

and for any $\mathbf{w} \in U^k$,

$$\mathbb{P}[\xi_{(a_1, \mathbf{w})} = \mathbf{u}] = \prod_{i=1}^k \mathbb{P}[\xi_{(a_1, \mathbf{w})} = u_i] = \left(\frac{1}{2k^2} \right)^k < \left(\frac{1}{k} \right)^k .$$

Therefore, one *always* has $[R^{(k)}]_{(a_1, \mathbf{w})}' = 0$ (see (1.4)), while by (5.2) and (2.3) the d_H distances between the random variables corresponding to a_1 and a_2 are well separated from zero. This establishes our claim at the start of this section regarding the nonexistence of an analogue of (3.2) from Theorem 3.2.

Intuitively, the reason this construction works is that we have selected range and parameter spaces whose size *depends* on the sequence length k . Note that we could have allowed $|U|$ to grow just linearly with k and still obtained the same conclusion. However, by allowing $|U|$ to grow more quickly with k our construction has a further notable property. Namely, the random variables corresponding to a_1 and a_2 become maximally distant under variation distance as $k \rightarrow \infty$, as inequality (5.2) reveals.

However, with mild extra conditions we can state a positive result. This positive result provides explicit bounds on the convergence of the MLE in the parametric setting.

THEOREM 5.1. *Assume $B = \{(a, \theta) : a \in A, \theta \in \Theta(a)\}$, and $\Xi : B \rightarrow U$ is a parametric random function, where A and U are finite sets. Assume that for a particular $(a, \theta) \in B$ there exists a $d_0 > 0$ such that for all $b \in A, b \neq a$, and $\theta' \in \Theta(b)$*

$$(5.3) \quad d((a, \theta), (b, \theta')) \geq d_0,$$

where d , as usual, denotes the variational distance. If the MLE is applied to invert the parametric random function $\Xi^{(k)} : A \rightarrow U^k$, which is a sequence of k independent trials of Ξ , then

$$(5.4) \quad \lim_{k \rightarrow \infty} [R^{(k)}]_{(a, \theta)}' = 1.$$

For a more precise result, set $U^+ = \{u \in U : \mathbb{P}[\xi_{(a, \theta)} = u] > 0\}$, and $\alpha = \min_{u \in U^+} \mathbb{P}[\xi_{(a, \theta)} = u]$. If

$$(5.5) \quad k > f(\alpha, d_0) \log \left(\frac{2|U^+|}{\epsilon} \right),$$

then MLE estimation returns a with probability at least $1 - \epsilon$, where

$$f(\alpha, d_0) = \max \left\{ \frac{16}{\alpha}, \frac{17 \log^2 \alpha (1 + \frac{2}{\alpha})^2}{\alpha d_0^4} \right\}.$$

Proof. For $u \in U$, define $p(u) = \mathbb{P}[\xi_{(a,\theta)} = u]$, and then $\alpha = \min_{u \in U^+} \{p(u)\} > 0$. Define $\hat{p}(u)$ as the corresponding relative frequency, i.e.,

$$(5.6) \quad \hat{p}(u) = \frac{1}{k} \#\{j : (\xi_j)_{(a,\theta)} = u\},$$

where ξ_j is the j th trial of the random function. Let $\delta = \frac{4}{\sqrt{17}}$, and let

$$\eta = \min \left\{ \frac{1}{2}, \frac{\delta d_0^2}{2|\log \alpha|(1 + \frac{2}{\alpha})} \right\}.$$

Then,

$$(5.7) \quad \eta |\log \alpha| + \frac{\eta |\log \alpha|}{\alpha(1 - \eta)} \leq \frac{\delta}{2} d_0^2.$$

By the large deviation inequality given in formula (14) of Appendix A in [1], we have

$$(5.8) \quad \mathbb{P}[|p(u) - \hat{p}(u)| > \eta p(u)] < 2e^{-c_\eta k p(u)},$$

where $c_\eta = \min\{-\log [e^\eta(1 + \eta)^{-(1+\eta)}], \frac{\eta^2}{2}\}$. Note that for $0 < \eta < 1/2$ we have $-\log[e^\eta(1 + \eta)^{-(1+\eta)}] \geq \frac{\eta^2(1-\eta)}{2}$ by Taylor expansion, and hence $c_\eta \geq \eta^2/4$. Therefore, formula (5.8) holds if we change c_η to $\eta^2/4$ in the exponent. Now suppose k satisfies inequality (5.5). Then,

$$k > \frac{4}{\alpha \eta^2} \log \left(\frac{2|U^+|}{\epsilon} \right)$$

by the definition of f and η . Consequently, $2|U^+|e^{-\eta^2 k \alpha/4} < \epsilon$, and so, with probability at least $1 - \epsilon$, we have

$$(5.9) \quad \forall u \in U \quad |p(u) - \hat{p}(u)| \leq \eta p(u).$$

(We also used the Bonferroni inequality, and the fact that, with probability 1, $p(u) = \hat{p}(u) = 0$ for all $u \in U \setminus U^+$.) For $x \in A, \omega \in \Theta(x)$, consider

$$(5.10) \quad L(x, \omega) = \sum_{u \in U} \hat{p}(u) \log \mathbb{P}[\xi_{x,\omega} = u].$$

(Here, as always in this kind of calculation, we use the convention $0 \times (-\infty) = 0$, which is supported by $\lim_{x \rightarrow 0^+} x \log x = 0$.) $L(x, \omega)$ is $\frac{1}{k}$ times the natural logarithm of the probability that the observed sequence of U -elements came from (x, ω) . Therefore $L(x, \omega) \leq 0$ is proportional to the log-likelihood of (x, ω) .

Now consider a fixed $b \in A, b \neq a$ and a fixed $\theta' \in \Theta(b)$. For $u \in U$, we use the notation $q(u) = \mathbb{P}[\xi_{(b,\theta')} = u]$.

We finish the proof conditional on the following event:

$$(5.11) \quad [(5.9) \text{ holds}] \text{ and } [u \notin U^+ \text{ implies } \hat{p}(u) = 0].$$

Note that the second part of the condition holds with probability 1, and so event (5.11) occurs with probability at least $1 - \epsilon$.

We distinguish two cases. In both cases we show

$$(5.12) \quad L(a, \theta) - L(b, \theta') > 0.$$

Since $L(a, \theta)$ (resp., $L(b, \theta')$) is the log-likelihood of getting the observed sequence from (a, θ) (resp., (b, θ')), (5.12) implies the correct reconstruction of a from the observed data by MLE by (1.4). Since this holds (with probability 1) for all θ' , conditional on event (5.11), and event (5.11) occurs with probability at least $1 - \epsilon$, the probability that MLE correctly reconstructs a will be at least $1 - \epsilon$, as required.

Case 1. There exists a $v \in U^+$ with $q(v) < \exp(\frac{\log \alpha}{\alpha(1-\eta)})$. In this case $L(b, \theta') \leq \hat{p}(v) \log q(v) < \log \alpha$, so $L(b, \theta') < \log \alpha$. On the other hand,

$$L(a, \theta) = \sum_{u \in U} \hat{p}(u) \log p(u) \geq \sum_{u \in U} \hat{p}(u) \log \alpha = \log \alpha.$$

Therefore, $L(a, \theta) > L(b, \theta')$.

Case 2. For all $u \in U^+$, $q(u) \geq \exp(\frac{\log \alpha}{\alpha(1-\eta)})$. We have, for all $u \in U^+$, $|\log q(u)| \leq \frac{|\log \alpha|}{\alpha(1-\eta)}$. Consider

$$(5.13) \quad \begin{aligned} L(a, \theta) - L(b, \theta') &= \sum_{u \in U} \hat{p}(u) \log \frac{p(u)}{q(u)} = \sum_{u \in U^+} \hat{p}(u) \log \frac{p(u)}{q(u)} \\ &= \sum_{u \in U} p(u) \log \frac{p(u)}{q(u)} + \sum_{u \in U^+} (\hat{p}(u) - p(u)) \log \frac{p(u)}{q(u)}. \end{aligned}$$

Notice that the first sum in (5.13) is exactly the Kullback–Leibler distance $d_{KL}((a, \theta), (b, \theta'))$. By formulae (2.8, 5.3) this first sum is at least $\frac{1}{2}d_0^2$. Since we condition on (5.9), $|\hat{p}(u) - p(u)| \leq \eta p(u)$. Hence, we can estimate the absolute value of the second sum in (5.13) by

$$(5.14) \quad \begin{aligned} \sum_{u \in U} \eta p(u) (|\log p(u)| + |\log q(u)|) &\leq \sum_{u \in U} \eta p(u) \left(|\log \alpha| + \frac{|\log \alpha|}{\alpha(1-\eta)} \right) \\ &= \eta |\log \alpha| + \frac{\eta |\log \alpha|}{\alpha(1-\eta)} \leq \frac{\delta}{2} d_0^2 \end{aligned}$$

by (5.7), and so $L(a, \theta) - L(b, \theta') > 0$. \square

Remarks.

1. Notice that, because $|U^+|\alpha \leq 1$, inequality (5.5) will hold whenever $k \geq f(\alpha, d_0) \log(\frac{2}{\alpha\epsilon})$. Notice that this bound on k (that suffices for parametric MLE to reconstruct a with probability at least $1 - \epsilon$) depends only on ϵ , d_0 , and α , and it is independent of the cardinality of A and U (cf. the bound we described for nonparametric MLE in the remark following Theorem 3.2).
2. Note also that the example described at the beginning of section 5 shows that one cannot strengthen Theorem 5.1 by simply dropping the role of α . That is, Theorem 5.1 fails if we replace (5.5) with the weaker condition that

$$k \geq f_1(d_0) \log \left(\frac{2|U^+|}{\epsilon} \right)$$

for some suitable function f_1 (that does not depend on α), since in the example described any such inequality will be satisfied for sufficiently large k ($|U|$ grows only quadratically with k), yet MLE fails to recover a_1 . A closer examination of this example shows that α converges to zero sufficiently fast with k for the bound in (5.5) to be violated.

3. Suppose that for each $b \in A$ we have (i) the set $\Theta(b)$ is a compact topological space, and (ii) the mapping from $\Theta(b)$ to the interval $[0, 1]$ defined by $(b, \theta) \mapsto \mathbb{P}(\xi_{(b, \theta)} = u)$ is continuous for each element $u \in U$. Then the separation property (5.3) required in Theorem 5.1 becomes equivalent to the (in general weaker) condition that for all $b \in A$, $b \neq a$, and $\theta' \in \Theta(b)$

$$(5.15) \quad d((a, \theta), (b, \theta')) > 0.$$

For example, for most models in the phylogenetic setting, assumptions (i) and (ii) will apply, and so MLE will be statistically consistent (that is, satisfy (5.4)), provided the model satisfies (5.15). In particular, the detailed analysis and additional assumption required by Chang [3] in order to establish (for a general Markov model on trees) a strengthening of (5.15) to the case $b = a$, $\theta \neq \theta'$ is unnecessary if one wishes simply to establish the statistical consistency of MLE in the estimation of a binary tree (and not the associated transition matrices of the model). There are also other models in use that satisfy (5.15) and thereby justify the statistical consistency of MLE. For example, consider a model in which sites evolve i.i.d. on a binary tree according to a stationary, reversible Markov process (with an unknown rate matrix) and with a rate factor (constant across the tree) drawn from a distribution \mathcal{D} . Such models satisfy (5.15) if \mathcal{D} is known and therefore the same for each possible tree [14, section 3.3]; however, (5.15) may fail if \mathcal{D} is unknown [13]. We note that Theorem 5.1 also provides the first explicit upper bounds on the sequence length required for MLE to accurately reconstruct a binary tree in the phylogenetic setting.

6. Conclusion and open problems. It would be interesting to see how much the bound on k given by Theorem 5.1 might be improved. This question applies both for the general setting in which Theorem 5.1 is stated, and also for more particular settings, such as arises in phylogeny.

In the general setting, observe that our upper bound on k given by Theorem 5.1 grows at the rate d_0^{-4} . Yet, in the nonparametric setting, if we let $d_0 = \min\{d(a, b) : a, b \in A, a \neq b\}$, then the analogous upper bound on k grows at the rate d_0^{-2} (by inequalities (3.5) and (2.3)). An interesting question is whether this discrepancy is essential in moving from the nonparametric to the parametric setting, or whether it can be avoided by a different argument. There are other significant differences between our results for the nonparametric and parametric setting—for example, although $|A|$ (but not $|U|$) enters directly into our bound on k in the nonparametric setting (inequality (3.5)), in the parametric setting $|A|$ is not directly mentioned but $|U^+|$ is.

Regarding more particular parametric MLE settings (such as in phylogeny) it is quite likely that the additional structure present in these instances may yield tighter bounds than those given by Theorem 5.1. It would be particularly desirable to set matching lower and upper bounds on the sequence length (the number of samples k) required by MLE in phylogeny reconstruction. It is clear that, for certain choices of the parameter θ , MLE may require longer sequences than other methods to correctly

reconstruct a phylogenetic tree (as discussed in [11] and [12]). Indeed, the statistical consistency of MLE in phylogeny was established only in 1996 by [3] in a result that, like Wald's earlier result [15], is based on a compactness argument that does not give an explicit bound on k . The significance of Theorem 5.1 is that it gives the first such explicit bounds for MLE, both in the phylogenetic setting and beyond.

6.1. Correction. Theorem 2.3 in [12], cited as Theorem 3.1 in our current paper, tacitly assumed $b \in N$. This assumption has to be made explicit.

Acknowledgments. The authors are indebted to Éva Czabarka for her invaluable comments on the manuscript. MAS also thanks Joe Chang for suggesting the usefulness of the Hellinger distance in the proof of Theorem 4.1. The authors are indebted for a number of useful comments from two anonymous referees.

REFERENCES

- [1] N. ALON AND J. H. SPENCER, *The Probabilistic Method*, John Wiley and Sons, New York, 1992.
- [2] G. CASELLA AND R. L. BERGER, *Statistical Inference*, Duxbury Press, Belmont, CA, 1990.
- [3] J. T. CHANG, *Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency*, *Math. Biosci.*, 137 (1996), pp. 51–73.
- [4] T. M. COVER AND J. A. THOMAS, *Elements of Information Theory*, John Wiley and Sons, New York, 1991.
- [5] P. L. ERDŐS, M. A. STEEL, L. A. SZÉKELY, AND T. WARNOW, *A few logs suffice to build (almost) all trees (I)*, *Random Structures Algorithms*, 14 (1999), pp. 153–184.
- [6] W. EVANS, C. KENYON, Y. PERES, AND L. J. SCHULMAN, *Broadcasting on trees and the Ising model*, *Ann. Appl. Probab.*, 10 (2000), pp. 410–433.
- [7] B. S. EVERITT, *The Cambridge Dictionary of Statistics*, Cambridge University Press, Cambridge, UK, 1998.
- [8] S. GUIASU, *Information Theory with Applications*, McGraw-Hill, New York, 1977.
- [9] L. LE CAM AND G. L. YANG, *Asymptotics in Statistics: Some Basic Concepts*, Springer-Verlag, New York, 1990.
- [10] J. NEYMAN, *Molecular studies of evolution: A source of novel statistical problems*, in *Statistical Decision Theory and Related Topics*, S. S. Gupta and J. Yackel, eds., Academic Press, New York, 1971, pp. 1–27.
- [11] M. STEEL AND D. PENNY, *Parsimony, likelihood, and the role of models in molecular phylogenetics*, *Mol. Biol. Evol.*, 17 (2000), pp. 839–850.
- [12] M. A. STEEL AND L. A. SZÉKELY, *Inverting random functions*, *Ann. Comb.*, 3 (1999), pp. 103–113.
- [13] M. A. STEEL, L. A. SZÉKELY, AND M. D. HENDY, *Reconstructing trees when sequence sites evolve at variable rates*, *J. Comput. Biol.*, 1 (1994), pp. 153–163.
- [14] C. TUFFLEY AND M. STEEL, *Modelling the covarion hypothesis of nucleotide substitution*, *Math. Biosci.*, 147 (1998), pp. 63–91.
- [15] A. WALD, *Note on the consistency of the maximum likelihood estimate*, *Ann. Math. Statistics*, 20 (1949), pp. 595–600.