stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 165/82 AUGUSTUS
(DEPARTMENT OF OPERATIONS RESEARCH)

A.J. HOFFMAN, A.W.J. KOLEN & M. SAKAROVITCH

TOTALLY-BALANCED AND GREEDY MATRICES

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classification: 05C50, 90C05, 90C10

Totally-balanced and greedy matrices*)

by

A.J. Hoffman**), A.W.J. Kolen***) & M. Sakarovitch****)

ABSTRACT

Totally-balanced and greedy matrices are (0,1)-matrices defined by excluding certain submatrices. For a $n\times m$ (0,1)-matrix A we show that the linear programming problem $\max \{by \mid yA \le c,\ 0 \le y \le d\}$ can be solved by a greedy algorithm for all $c \ge 0$, $d \ge 0$ and $b_1 \ge b_2 \ge \dots \ge b_n \ge 0$ if and only if A is a greedy matrix. Furthermore we show constructively that if b is integer, then the corresponding primal problem $\min\{cx + dz \mid Ax + z \ge b,\ x \ge 0,\ z \ge 0\}$ has an integer optimal solution. A polynomial-time algorithm is presented to transform a totally-balanced matrix into a greedy matrix as well as to recognize a totally-balanced matrix. This transformation algorithm together with the result on greedy matrices enables us to solve a class of integer programming problems defined on totally-balanced matrices. Two examples arising in tree location theory are presented.

KEY WORDS & PHRASES: *Integer programming, greedy algorithm, totally-balanced matrices, location theory*

---

*) This report is submitted for publication in SIAM Journal on Algebraic and Discrete Methods.

**) I.B.M., Yorktown Heights, NY, U.S.A.

***) Erasmus Universiteit, Rotterdam, The Netherlands; this author's research was done at the Mathematisch Centrum Amsterdam. The Netherlands.

****) I.M.A.G., Grenoble, France

1. INTRODUCTION

A (0,1)-matrix is *balanced* if it does not contain an odd square submatrix of size at least three with all row and column sums equal to two. Balanced matrices have been studied extensively by BERGE [3] and FULKERSON et al. [7]. We consider a more restrictive class of matrices called totally-balanced (LOVÁSZ [11]). A (0,1)-matrix is *totally-balanced* if it does not contain a square submatrix of size at least three which has no identical columns and its row and column sums equal to two.

EXAMPLE 1.1. Let $T = (V,E)$ be a tree with vertex set $V = \{v_1,v_2,\dots,v_n\}$ and edge set E. Each edge $e \in E$ has a positive length $\ell(e)$. A *point* on the tree can be a vertex or a point anywhere along the edge. The *distance* $d(x,y)$ between the two points x and y on T is defined as the length of the path between x and y. A *neighborhood subtree* is defined as the set of all points on the tree within a given distance (called *radius*) of a given point (called *center*). Let x_i $(i = 1,2,\dots,m)$ be points on T and let r_i $(i = 1,2,\dots,m)$ be nonnegative numbers. Define the neighborhood subtrees T_i by $T_i = \{y \in T \mid d(y,x_i) \le r_i\}$. Let $A = (a_{ij})$ be the n×m (0,1)-matrix defined by $a_{ij} = 1$ if and only if $v_i \in T_j$ It was first proved by GILES [8] that A is totally-balanced. This result was generalized by TAMIR [12]: Let Q_i $(i = 1,2,\dots,n)$ and R_j $(j = 1,2,\dots,m)$ be neighborhood subtrees and let the n×m (0,1)-matrix $B = (b_{ij})$ be defined by $b_{ij} = 1$ if and only if $Q_i \cap R_j \neq \emptyset$. Then B is totally balanced. □

Motivation for the types of problems to be studied in this paper is given by the following two examples from tree location theory stated in Example 1.2.

EXAMPLE 1.2. Let $T = (V,E)$ be a tree, let T_j $(j = 1,2,\dots,m)$ be neighborhood subtrees and let $A = (a_{ij})$ be the (0,1)-matrix as defined in Example 1.1. We interpret x_j as the possible location of a facility, and T_j as the service area of a facility at x_j, i.e., x_j can only serve clients located at T_j (we assume clients to be located at vertices). We assume there is a cost c_j associated with establishing a facility at x_j $(j = 1,2,\dots,m)$. The *minimum cost covering problem* is to serve all clients at minimum cost.

This problem can be formulated as

$$(1.3)\qquad \begin{aligned} &\min \sum_{j=1}^{m} c_j x_j \\ &\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1,2,\dots,n, \\ &\qquad\quad x_j \in \{0,1\},\ j = 1,2,\dots,m. \end{aligned}$$

Let us relax the condition in this problem that each client has to be served by assuming that if a client located at vertex v_i is not served by a facility, then a penalty cost of d_i $(i = 1,2,\dots,n)$ is charged. The *minimum cost operating problem* is to minimize the total cost of establishing facilities and not serving clients, i.e.,

$$(1.4)\qquad \begin{aligned} &\min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} d_i z_i \\ &\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j + z_i \geq 1,\ i = 1,2,\dots,n, \\ &\qquad\quad x_j \in \{0,1\},\quad j = 1,2,\dots,m, \\ &\qquad\quad z_i \in \{0,1\},\quad i = 1,2,\dots,n. \quad \square \end{aligned}$$

Let $A = (a_{ij})$ be a (0,1)-matrix. We can associate a subset of rows to each column, namely those rows which have a one in this column. An $n\times m$ (0,1)-matrix is called greedy if for all $i = 1,2,\dots,n$ the following holds; all columns having a 1 in row i can be totally ordered by inclusion when restricted to the rows $i,i+1,\dots,n$. An equivalent definition is to say that the two 3×2 submatrices

$$(1.5)\qquad \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

do not occur.

Why the name "greedy" is chosen will become clear in the next paragraph. It is a trivial observation that each greedy matrix is totally balanced. We will prove in Section 3 that, conversely, the rows of a totally-balanced matrix can be permuted in such a way that the resulting matrix is greedy. The proof will be constructive.

Let the n×m(0,1)-matrix $A = (a_{ij})$ be greedy. Consider the problem (P) given by

$$\text{(P)} \qquad \begin{aligned} \min \ & \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} d_i z_i \\ \text{s.t.} \ & \sum_{j=1}^{m} a_{ij} x_j + z_i \geq b_i, \quad i = 1,2,\ldots,n, \\ & x_j \geq 0, \quad j = 1,2,\ldots,m, \\ & z_i \geq 0, \quad i = 1,2,\ldots,n. \end{aligned}$$

The dual problem (D) is given by

$$\text{(D)} \qquad \begin{aligned} \max \ & \sum_{i=1}^{n} b_i y_i \\ \text{s.t.} \ & \sum_{i=1}^{n} y_i a_{ij} \leq c_j, \quad j = 1,2,\ldots,m, \\ & 0 \leq y_i \leq d_i, \quad i = 1,2,\ldots,n. \end{aligned}$$

We will show in Section 2 that problem (D) can be solved by a greedy algorithm for all $c \geq 0$, $d \geq 0$ and $b_1 \geq b_2 \geq \ldots \geq b_n \geq 0$ if and only if the matrix A is greedy. Further we construct an optimal solution to the primal problem (P) which has the property that it is an integer solution whenever b is integer. This means that after we use the algorithm of Section 3 to transform a totally-balanced matrix into a greedy matrix we can solve the two location problems using the result of Section 2.

After we submitted the first version of the paper we found out about the work done by FARBER. FARBER [5,6] studies strongly chordal graphs and gives polynomial-time algorithms to find a minimal weighted dominating set and minimal weighted independent dominating set. In these algorithms FARBER uses the same approach as described in Section 2. In another paper ANSTEE & FARBER [1] relate strongly chordal graphs to totally-balanced matrices. This paper contains the relationship between totally-balanced and greedy matrices described in Section 3 as well as a recognition algorithm for a totally-balanced matrix which, however, is less efficient than the one described here.

2. THE ALGORITHM

A greedy (0,1)-matrix is in *standard greedy form* if for all i,j,k,ℓ with $i < j$, $k < \ell$ the equalities $a_{ik} = a_{i\ell} = a_{jk} = 1$ imply that $a_{j\ell} = 1$. In Section 3 we will show that by a permutation of the columns an $n\times m$ greedy matrix can be transformed into a matrix in standard greedy form in $O(mn)$ time. The algorithm of Section 3 applied to a totally-balanced matrix also produces a matrix in standard greedy form. In this section we will assume that the greedy matrix is in standard greedy form. This assumption does not affect the dual solution obtained but facilitates the description of the primal solution.

Let $A = (a_{ij})$ be an $n\times m$ (0,1)-matrix in standard greedy form, let c_j $(j = 1,2,\dots,m)$ and d_i $(i = 1,2,\dots,n)$ be positive numbers (the case when one of these numbers is zero can be treated similarly) and let $b_1 \geq b_2 \geq \dots \geq b_n \geq 0$. A feasible solution $\bar{y}$ of problem (D) is obtained by a greedy algorithm. The values of $\bar{y}_i$ are determined in order of increasing i and taken to be as large as possible. A constraint j is *tight* if $\sum_{i=1}^{n} \bar{y}_i\, a_{ij} = c_j$. The index $\alpha(j)$ denotes the largest index of a positive $\bar{y}$-value in the tight constraint j, J denotes a set of tight constraints. The greedy procedure is formulated in Algorithm D.

<u>ALGORITHM D</u>

```
begin $J:=\emptyset;\hat{c}:=c;$
      for $i:=1$ step 1 to n
      do $\bar{y}_i:=\min\{d_i,\min_{j:a_{ij}=1}\{\hat{c}_j\}\};$
         if $\bar{y}_i > 0$ then if $\bar{y}_i = \hat{c}_j$ for some $j$ then choose the largest $j$
                                  and let $J:=J\cup\{j\}$; $\alpha(j):=i$
                            fi;
                            $\hat{c}_j:=\hat{c}_j - \bar{y}_i$ for all $j$ such that $a_{ij} = 1$
         fi
      od
end
```

For the solution $\bar{y}$ constructed by Algorithm D the following holds:

(2.1) If $\bar{y}_k = d_k$, then either there is no $j \in J$ such that $a_{kj} = 1$ or there is a $j \in J$ such that $a_{kj} = 1$ and $\alpha(j) \geq k$,

and

(2.2) If $\bar{y}_k = 0$, then there is a $j \in J$ such that $a_{kj} = 1$ and $\alpha(j) < k$.

Property (2.1) follows immediately from the algorithm. If $\bar{y}_k = 0$, then there exists an index i, $i < k$ and a constraint $\hat{j}$ such that constaint $\hat{j}$ is tight, $a_{i\hat{j}} = a_{k\hat{j}} = 1$, and i is the largest index of a positive $\bar{y}$-value in constraint $\hat{j}$. During the iteration in which $\bar{y}_i$ was determined we have added an index $j \geq \hat{j}$ with $\alpha(j) = i$ to J. Since A is a standard greedy form we have $a_{kj} = 1$. This proves Property (2.2).

EXAMPLE 2.3. The matrix and costs of the example as well as the results of Algorithm D are given in Figure 2.4. We assume $d_i = 2$ $(i = 1,\dots,9)$ and $(b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8,b_9) = (6,5,4,3,3,2,2,1)$.

$$\hat{c}_1 = 3,\ \hat{c}_2 = 4,\ \hat{c}_3 = 5,\ \hat{c}_4 = 2,\ \hat{c}_5 = 3,\ \hat{c}_6 = 5,\ \hat{c}_7 = 3.$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{array}{l} \bar{y}_1 = 2,\ \hat{c}_1 = 1,\ \hat{c}_2 = 2. \\ \bar{y}_2 = 1,\ J = \{1\},\ \alpha(1) = 2,\ \hat{c}_1 = 0,\ \hat{c}_2 = 1. \\ \bar{y}_3 = 0. \\ \bar{y}_4 = 2,\ \hat{c}_3 = 3.\ \hat{c}_6 = 3. \\ \bar{y}_5 = 2,\ \hat{c}_3 = 1,\ \hat{c}_6 = 1. \\ \bar{y}_6 = 2,\ J = \{1,4\},\ \alpha(4) = 6,\ \hat{c}_4 = 0,\ \hat{c}_7 = 1. \\ \bar{y}_7 = 1,\ J = \{1,4,7\},\ \alpha(7) = 7,\ \hat{c}_2 = 0,\ \hat{c}_5 = 2,\ \hat{c}_7 = 0. \\ \bar{y}_8 = 0. \\ \bar{y}_9 = 0. \end{array}$$

Figure 2.4. Example of Algorithm D.

The value of the feasible dual solution $\bar{y}$ is 35. □

The primal solution $\bar{x},\bar{z}$ is constructed by Algorithm P with has as input the set of tight constraints J and the indices $\alpha(j)$ $(j \in J)$.

ALGORITHM P

```
begin $\hat{b}:=b; \bar{x}_j:=0$ for all $j \notin J$;
    while $J \neq \emptyset$
    do (let k be the last column of J)
        $\bar{x}_k:=\hat{b}_{\alpha(k)}$;
        $\hat{b}_i:=\hat{b}_i - \bar{x}_k$ for all i such that $a_{ik}=1$;
        $J:=J\backslash\{k\}$
    od;
    for i:=1 step 1 to n do $\bar{z}_i:=\max(0,\hat{b}_i)$ od
end
```

EXAMPLE 2.5. Apply Algorithm P to Example 2.3.
$\bar{x}_2 = \bar{x}_3 = \bar{x}_5 = \bar{x}_6 = 0$, $\hat{b} = b$.
Iteration 1: $\bar{x}_7 = 2$, $\hat{b}_6 = \hat{b}_7 = \hat{b}_8 = 0$, $\hat{b}_9 = -1$.
Iteration 2: $\bar{x}_4 = 0$.
Iteration 3: $\bar{x}_1 = 5$, $\hat{b}_1 = 1$, $\hat{b}_2 = 0$, $\hat{b}_3 = -1$.
$\bar{z}_1 = 1$, $\bar{z}_4 = \bar{z}_5 = 3$, all other $\bar{z}_i$ values are zero.
It is easy to check that $\bar{x},\bar{z}$ is a feasible primal solution with value 35. Since the values of the feasible primal and dual solutions are equal they are both optimal. □

If we prove that $\bar{x}_j \geq 0$ for all $j \in J$, then it is clear that $\bar{y}$ and $\bar{x},\bar{z}$ are feasible solutions. In order to prove that they are optimal solutions we show that the complementary slackness relations of linear programming hold. These conditions are given by

(2.6) $$\bar{x}_j\Big(\sum_{i=1}^{n} \bar{y}_i a_{ij} - c_j\Big) = 0, \qquad j = 1,2,\ldots,m,$$

(2.7) $$\bar{y}_i\Big(\sum_{j=1}^{m} a_{ij}\bar{x}_j + \bar{z}_i - b_i\Big) = 0, \quad i = 1,2,\ldots,n,$$

(2.8) $$\bar{z}_i(\bar{y}_i - d_i) = 0, \qquad i = 1,2,\ldots,n.$$

Let us denote by $\hat{J}$ the set of column indices in Algorithm P which is initially equal to J and decreases by one element at each iteration. Accordingly let $b_i(\hat{J}) = b_i - \sum_{j\in J\backslash\hat{J}} a_{ij}\bar{x}_j$, $i = 1,2,\ldots,n$. Define I by $I = \{i \mid \exists j \in J\ \alpha(j) = i\}$.

The following properties hold for Algorithm P.

PROPERTY 2.9. If $a_{ij} = a_{\ell j} = 1$, $i < \ell$, $j \in \hat{J}$, then $b_i(\hat{J}) \geq b_\ell(\hat{J})$.

PROOF. This is true at the start of the algorithm since $b_i \geq b_\ell$, $i < \ell$. Let k be the last column of $\hat{J}$. Property 2.9 could be altered only if $a_{ik} = 1$ and $a_{\ell k} = 0$, which is rules out by the fact that A is in standard greedy form. □

PROPERTY 2.10. $b_i(\hat{J}) \geq 0$ for all $i \in I$.

PROOF. This is true at the start of the algorithm since $b_i \geq 0$. Let k be the last column of $\hat{J}$. Using Property 2.9 we know that Property 2.10 could be altered only if $a_{ik} = 1$ and $i > \alpha(k)$, which is ruled by definition of $\alpha(k)$. □

PROPERTY 2.11. $b_i(\emptyset) = 0$ for all $i \in I$.

PROOF. Let $i \in I$. There exists a $j \in J$ such that $\alpha(j) = i$. At the iteration at which j was the last column of $\hat{J}$ we define $\bar{x}_j = b_i(\hat{J})$ and hence after this iteration we have $b_i(\hat{J}) = 0$. Combining this with Property 2.10 we get $b_i(\emptyset) = 0$. □

PROPERTY 2.12. If $\bar{y}_k > 0$, $k \notin I$, then $\sum_{j \in J} a_{kj}\bar{x}_j \leq b_k$.

PROOF. If $\bar{y}_k > 0$, $k \notin I$, then according to (2.1) we have to consider two cases:

1. There is no $j \in J$ such that $a_{kj} = 1$. In this case we have $\sum_{j \in J} a_{kj}\bar{x}_j = 0 \leq b_k$;
2. There is a $j \in J$ such that $a_{kj} = 1$ and $\alpha(j) > k$ (note that since $k \notin I$ we can rule out $\alpha(j) = k$). Using Property 2.9 and 2.11 we get $b_k(\emptyset) \geq b_{\alpha(j)}(\emptyset) = 0$. □

PROPERTY 2.13. If $\bar{y}_k = 0$, then $\sum_{j \in J} a_{kj}\bar{x}_j \geq b_k$.

PROOF. If $\bar{y}_k = 0$, then according to (2.2) there exists a $j \in J$ such that $a_{kj} = 1$ and $\alpha(j) < k$. Using Property 2.9 and 2.11 we get $b_k(\emptyset) \leq b_{\alpha(j)}(\emptyset) = 0$. □

It follows from Property 2.10 that $\bar{x}_j \geq 0$ for all $j \in J$. Hence $\bar{x},\bar{z}$ is a feasible solution. For the complementary slackness relations (2.6) follows by construction, (2.7) and (2.8) follow from Property 2.11, 2.12 and 2.13.

THEOREM 2.14. *Problem* (D) *is solved by Algorithm* D *for all* $c \geq 0$, $d \geq 0$ *and* $b_1 \geq b_2 \geq \ldots \geq b_n \geq 0$ *if and only if* A *is greedy.*

PROOF. If A is greedy, then we transform A into standard greedy form as indicated in Section 3 by a permutation of the columns. This permutation does not effect the dual solution, which was shown to be optimal.

If A is not greedy, then there exists a 3×2 submatrix of the form
$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$.
Let the rows be given by $i_1 < i_2 < i_3$ and columns by $j_1 < j_2$.
Set $d_i = 0$ for all $i \notin \{i_1,i_2,i_3\}$, $d_{i_1} = d_{i_2} = d_{i_3} = 1$. $c_j = 3$ for all j except $c_{j_1} = c_{j_2} = 1$, $b_i = 1$ for all $i = 1,2,\ldots,n$. If we apply Algorithm D we get $\bar{y}_{i_1} = 1$, all other $\bar{y}_i$ are zero. The value of this solution is 1. However $\bar{y}_{i_2} = \bar{y}_{i_3} = 1$ and all other $\bar{y}_i$ are zero is a feasible solution with value 2. This shows that Algorithm D does not solve this instance of problem (D). □

3. THE STANDARD GREEDY FORM TRANSFORMATION

In this section we show that an n×m greedy matrix can be transformed into standard greedy form by a permutation of the columns in O(nm) time. We give an $O(nm^2)$ algorithm which transforms an n×m totally-balanced matrix into standard greedy form as well as an $O(nm^2)$ algorithm which recognizes an n×m totally-balanced matrix. Since a matrix is in standard greedy form if and only if its transpose is in standard greedy form we may assume without loss of generality that $m \leq n$.

Let $A = (a_{ij})$ be an n×m totally-balanced matrix without zero rows or columns. We consider column j of A as a subset of rows, namely those rows which are covered by column j. Let us denote column j by E_j. Then $E_j = \{i \mid a_{ij} = 1\}$. Let the matrix A be given by its columns $E_1,E_2,\ldots,E_m$. The algorithm produces a 1-1 mapping $\sigma : \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ corresponding to a transformation of the rows of A ($\sigma(i) = j$ indicates that row i becomes row j in the transformed matrix) and a 1-1 mapping $\tau : \{E_1,\dots,E_m\} \to \{1,\dots,m\}$ corresponding to a transformation of the columns of A ($\tau(E_i) = j$ implies that column i becomes column j in the transformed matrix). We present the algorithm in an informal way and give an example to demonstrate it.

The algorithm consists of m iterations. At iteration i we determine the column E for which $\tau(E) = m-i+1$ $(1 \le i \le m)$. At the beginning of each iteration the rows are partitioned into a number of groups, say $G_r,\dots,G_1$. If $i < j$, then for all $k \in G_i$ and $\ell \in G_j$ we have $\sigma(k) < \sigma(\ell)$, i.e., rows belonging to G_i precede rows belonging to G_j in the transformed matrix. Rows b and c belong to the same group G at the beginning of iteration i if and only if for all columns E we have determined so far, i.e., all columns E for which $\tau(E) \ge m-i+2$, we cannot distinguish between the rows b and c, i.e., $b \in E$ if and only if $c \in E$. At the beginning of iteration 1 all rows belong to the same group. Let $G_r,\dots,G_1$ be the partitioning into groups at the beginning of iteration i $(1 \le i \le m)$. For each column E not yet determined we calculate the vector d_E of length r, where $d_E(j) = |G_{r-j+1} \cap E|$ $(j = 1,2,\dots,r)$. A column E for which d_E is a lexicographically largest vector is the column determined at iteration i with $\tau(E) = m-i+1$. After we have determined E we can distinguish between some rows in the same group G if $1 \le |G \cap E| < |G|$. If this is the case we shall take rows in $G \backslash E$ to precede rows in $G \cap E$ in the transformed matrix. This can be expressed by adjusting the partitioning into groups in the following way. For $j = r,r-1,\dots,1$ respectively we check if the intersection of G_j and E is not empty and not equal to G_j. If this is the case we increase the index of all groups with index greater than j by one and partition the group G_j into two groups called G_j and G_{j+1}, $G_{j+1} = G_j \cap E$ and $G_j = G_j \backslash E$. The algorithm ends after m iterations with a partitioning into groups, say $G_r,\dots,G_1$. The permutation σ is defined by $\sigma(k) < \sigma(\ell)$ if $k \in G_i$ and $\ell \in G_j$, $i < j$. Within a group G_i we assign the values $\sum_{j=1}^{i-1} |G_j| + 1,\dots,\sum_{j=1}^{i} |G_j|$ in an arbitrary way to the elements in this group. The number of computations we have to do at each iteration is $O(mn)$. Therefore the time complexity of this algorithm is $O(nm^2)$.

<u>EXAMPLE 3.1</u>. The 9×7 (0,1)-matrix A is given by its columns $E_1 = \{1,2,3\}$, $E_2 = \{1,2,3,5\}$, $E_3 = \{4,5\}$, $E_4 = \{3,4,5,9\}$, $E_5 = \{5,8,9\}$, $E_6 = \{6,7,8,9\}$, $E_7 = \{6,7,8\}$.

Iteration 1: $G_1 = (1,2,3,4,5,6,7,8,9)$.
$d_{E_i} = (|E_i|)$, choose E_4, $\tau(E_4) = 7$.

Iteration 2: $G_2 = (3,4,5,9)$, $G_1 = (1,2,6,7,8)$.

E	E_1	E_2	E_3	E_5	E_6	E_7
d_E	(1,2)	(2,2)	(2,0)	(2,1)	(1,3)	(0,3)

Choose E_2, $\tau(E_2) = 6$.

Iteration 3: $G_4 = (3,5)$, $G_3 = (4,9)$, $G_2 = (1,2)$, $G_1 = (6,7,8)$.

E	E_1	E_3	E_5	E_6	E_7
d_E	(1,0,2,0)	(1,1,0,0)	(1,1,0,1)	(0,1,0,3)	(0,0,0,3)

Choose E_5, $\tau(E_5) = 5$.

Iteration 4: $G_7 = (5)$, $G_6 = (3)$, $G_5 = (9)$, $G_4 = (4)$, $G_3 = (1,2)$, $G_2 = (8)$, $G_1 = (6,7)$.

E	d_E
E_1	(0,1,0,0,2,0,0)
E_3	(1,0,0,1,0,0,0)
E_6	(0,0,1,0,0,1,2)
E_7	(0,0,0,0,0,1,2)

Choose E_3, $\tau(E_3) = 4$.

From now on the groups do not change. Therefore $\tau(E_1) = 3$, $\tau(E_6) = 2$, $\tau(E_7) = 1$. A mapping σ is given by $\sigma : (6,7,8,1,2,4,9,3,5) \to (1,2,3,4,5,6,7,8,9)$. The mapping τ is given by $\tau : (E_7,E_6,E_1,E_3,E_5,E_2,E_4) \to (1,2,3,4,5,6,7)$. The transformed matrix is the one used in Example 2.3. □

A mapping $\sigma : \{1,2,\dots,n\} \to \{1,2,\dots,n\}$ is a *nest ordering with respect to* $E_1,E_2,\dots,E_m$ if all columns covering the row j defined by $\sigma(j) = i$ can be totally ordered by inclusion when restricted to the rows k of the matrix with $\sigma(k) \geq i$ $(i = 1,2,\ldots,n)$. By a *lexicographic ordering* of subsets $E_1,E_2,\ldots,E_m$ of $\{1,2,\ldots,n\}$ the following is meant. With each set E we associate a vector b_E of length $|E|$. The first component of b_E is the largest element of E, the second component is the second largest element, and so on. E_i is lexicographically less than or equal to E_j if b_{E_i} is lexicographically less than or equal to b_{E_j}. Ties, which only occur when two subsets contain the same elements, are broken arbitrarily. We can order $E_1,E_2,\ldots,E_m$ in a lexicographically nondecreasing order in $O(mn)$ time (see [2]).

LEMMA 3.2. *Let A be a $(0,1)$-matrix such that the ordering of the rows forms a nest ordering with respect to the columns, and the columns are ordered in a lexocigraphically nondecreasing order. Then the matrix A is in standard greedy form.*

PROOF. Suppose $a_{ik} = a_{i\ell} = a_{jk} = 1$, $i < j$, $k < \ell$. Since $i \in E_k \cap E_\ell$ we have that $E_k^i \subseteq E_\ell^i$ or $E_\ell^i \subseteq E_k^i$, where $E_k^i = E_k \setminus \{1,\ldots,i-1\}$ and $E_\ell^i = E_\ell \setminus \{1,\ldots,i-1\}$. Since E_k is lexicographically less than or equal to E_ℓ we have $E_k^i \subseteq E_\ell^i$. Hence $a_{jk} = 1$ implies that $a_{j\ell} = 1$. □

Lemma 3.2 shows that if we order the columns of a greedy matrix in a lexicographically nondecreasing order, then the matrix is in standard greedy form.

Let σ and τ be the mapping constructed by the algorithm applied to a totally-balanced matrix. We shall prove that σ is a nest ordering with respect to the columns and τ is a lexicographic ordering of the columns after the rows have been reordered according to σ. Two columns E_1,E_2 are *comparable* if $E_1 \subseteq E_2$ or $E_2 \subseteq E_1$; they are *incomparable* if they are not comparable.

LEMMA 3.3. *Let E_1,E_2 be incomparable columns such that $\tau(E_1) < \tau(E_2)$, let $i \in E_1 \setminus E_2$ and let j be the largest element with respect to σ in $E_2 \setminus E_1$, i.e., there is no $k \in E_2 \setminus E_1$ such that $\sigma(k) > \sigma(j)$. Then $\sigma(i) < \sigma(j)$.* 11

PROOF. Consider the iteration at which E_2 was determined. Let $G_r,\ldots,G_1$ be the partitioning into groups at the beginning of this iteration. Let k be the largest index for which $G_k \cap E_1 \neq G_k \cap E_2$. Then $j \in G_k$. If $i \in G_f$ with $f < k$, then $\sigma(i) < \sigma(j)$. If $i \in G_k$, then after E_2 is determined the group G_k is partitioned into two groups $G_k \cap E_2$ and $G_k \setminus E_2$, where rows in $G_k \setminus E_2$

precede rows in $G_k \cap E_2$ in the transformed matrix. Since $i \in G_k \setminus E_2$ and $j \in G_k \cap E_2$ we have $\sigma(i) < \sigma(j)$. □

It follows from Lemma 3.3 that the algorithm produces a lexicographically nondecreasing ordering of the columns. It remains to show that the mapping σ constructed by the algorithm applied to a totally-balanced matrix is a nest ordering with respect to the columns. Suppose σ is not a nest ordering with respect to the columns and let i_0 be the largest row with respect to σ for which there exist two incomparable columns E_1 and E_2 with respect to all rows i with $\sigma(i) \geq \sigma(i_0)$. Without loss of generality assume $\tau(E_1) < \tau(E_2)$. Let i_1 be the largest element with respect to σ in $E_1 \setminus E_2$ and let i_2 be the largest element with respect to σ in $E_2 \setminus E_1$. It follows from Lemma 3.3 that $\sigma(i_2) > \sigma(i_1)$. We call (i_0,i_1,i_2,E_1,E_2) a *2-chain*. We generalize the definition of a 2-chain to a m-chain using the following definition. A column E_k separates i from j if $i \in E_k$, $j \notin E_k$ and for all columns E_ℓ with $\tau(E_\ell) > \tau(E_k)$ we have $i \in E_\ell$ if and only if $j \in E_\ell$. Note that if $\sigma(i) > \sigma(j)$ and i and j are not covered by the same columns, then there is a column E which separates i from j. This can be seen as follows. Consider the last iteration at which i and j are in the same group and let E be the column determined at this iteration. Since $\sigma(i) > \sigma(j)$ we have that after this iteration i is in a group with larger index than the group containing j. This implies $i \in E$ and $j \notin E$ and therefore E separates i from j. We call $(i_0,i_1,i_2,\ldots,i_m,E_1,E_2,\ldots,E_m)$ a m-*chain* $(m \geq 2)$ if

1. $i_j \in E_k \Leftrightarrow j = k$ or $j = k-2$ $(k = 1,2,\ldots,m)$, where $i_{-1} = i_0$,
2. $\sigma(i_{j+1}) > \sigma(i_j)$ $(j = 0,1,\ldots,m-1)$,
3. $\tau(E_{j+1}) > \tau(E_j)$ $(j = 1,2,\ldots,m-1)$,
4. E_j separates i_{j-2} from i_{j-3} $(j = 3,\ldots,m)$,
5. i_j is the largest row with respect to σ in E_j which is not contained in E_{j-1} $(j = 1,2,\ldots,m)$, where $E_0 = E_2$.

<u>THEOREM 3.4.</u> *A m-chain can be extended to a* m+1-*chain* $(m \geq 2)$.

<u>PROOF.</u> Since $\sigma(i_{m-1}) > \sigma(i_{m-2})$ and i_{m-2} and i_{m-1} are not covered by the same columns $(i_{m-1} \notin E_m)$ we have that there exists a column E_{m+1} which separates

i_{m-1} from i_{m-2}. Note that by definition $i_{m-2} \notin E_{m+1}$ and $\tau(E_{m+1}) > \tau(E_m)$. E_m separates i_{m-2} from i_{m-3}. Since $i_{m-2} \notin E_{m+1}$ and $\tau(E_{m+1}) > \tau(E_m)$ we have $i_{m-3} \notin E_{m+1}$. Repeating this argument for $E_{m-1},\ldots, E_3$ respectively shows that $i_{m-2},\ldots,i_0 \notin E_{m+1}$. If $i_m \in E_{m+1}$, then the rows $i_0,\ldots,i_m$ and columns $E_1,\ldots,E_{m+1}$ define a square submatrix of size $m+1 \geq 3$ with no identical columns and all its row and column sums equal to two. This contradicts the fact that A is totally-balanced. Hence $i_m \notin E_{m+1}$. Since $i_m \notin E_{m+1}$ and $i_{m-1} \notin E_m$ we have that E_m and E_{m+1} are incomparable. Let i_{m+1} be the largest row with respect to σ in $E_{m+1} \setminus E_m$. It follows from Lemma 3.3 that $\sigma(i_{m+1}) > \sigma(i_m)$. In order to prove that the m-chain extended with i_{m+1} and E_{m+1} is a m+1 - chain we have to prove that $i_{m+1} \notin E_k$ for $k = 1,\ldots,m$. We already know that $i_{m+1} \notin E_m$. Suppose $i_{m+1} \in E_k$ for some k ($1 \leq k \leq m$). Let k be the index such that $i_{m+1} \in E_k$ and $i_{m+1} \notin E_{k+1}$. Note that under the assumption made such an index exists. Since i_k is the largest row with respect to σ in $E_k \setminus E_{k-1}$ and $\sigma(i_{m+1}) > \sigma(i_k)$ we have $i_{m+1} \in E_{k-1}$. If $k = 1$, then this contradicts the fact that $i_{m+1} \notin E_2 = E_0$. If $k > 1$, then we have $i_{m+1} \in E_{k-1} \setminus E_{k+1}$ and $i_{k+1} \in E_{k+1} \setminus E_{k-1}$ which contradicts the fact that by definition of i_0 all columns covering i_{k-1} (for example, E_{k-1} and E_{k+1}) are comparable with respect to all rows i with $\sigma(i) \geq \sigma(i_{k-1})$ (Note that $\sigma(i_{k-1}) > \sigma(i_0)$ and $\sigma(i_{m+1}) > \sigma(i_{k+1}) > \sigma(i_{k-1})$). We conclude that $i_{m+1} \notin E_k$ for all $k = 1,2,\ldots,m$. □

<u>COROLLARY 3.5</u>. *The mapping σ constructed by the algorithm applied to a totally-balanced matrix is a nest ordering with respect to the columns.*

<u>PROOF</u>. If σ is not a nest ordering, then there exists a 2-chain which according to Theorem 3.4 can be extended infinitely. Since the number of rows is finite it follows that σ is a nest ordering. □

This completes the correctness proof of the algorithm. The following theorem shows how we can recognize an n×m totally-balanced matrix in $O(nm^2)$ time using the mapping σ constructed by the transformation algorithm.

<u>THEOREM 3.6</u>. *A (0,1)-matrix A is totally-balanced if and only if the mapping σ constructed by the transformation algorithm applied to A is a nest ordering.*

PROOF. If A is totally-balanced, then we proved that σ is a nest ordering. If A is not totally-balanced, then there is a square submatrix A_1 of size at least three with no identical columns and row and column sums equal to two. Let i_1 be the smallest row with respect to σ of A_1, and let E_j and E_k be the two columns of A_1 covering this row. Let i_2 and i_3 be the other rows of A_1 covered by E_j and E_k respectively. It follows that $i_2 \in E_j \setminus E_k$ and $i_3 \in E_k \setminus E_j$, i.e., E_j and E_k are not comparable with respect to all rows i with $\sigma(i) \geq \sigma(i_1)$. Hence σ is not a nest ordering.

We can find σ in $O(nm^2)$ time. Checking whether σ is a nest ordering can be done by comparing all columns covering the row j defined by $\sigma(j) = i$ for $i = 1,2,\ldots n$, respectively. Columns which have been compared because they cover a row j with $\sigma(j) = i$ do not have to be compared at any other iteration k with $k > i$. So we only have to check each pair of columns at most once. This can be done in $O(nm^2)$-time. Hence the recognition also requires $O(nm^2)$-time.

In a previous paper (BROUWER & KOLEN [4], see also KOLEN [10]) it was shown that there exists a row of a totally-balanced matrix such that all columns covering this row can be totally ordered by inclusion. As indicated by one of the referees this result can be used to derive an $O(n^2m)$ algorithm to transform a totally-balanced matrix into standard greedy form as compared to the $O(nm^2)$ algorithm presented (note $m \leq n$). The algorithm presented is a constructive proof of the fact that a nest ordering exists.

We finish discussing the relationship between totally-balanced matrices and chordal bipartite graphs. A *chordal bipartite graph* is a bipartite graph for which every cycle of length strictly greater than four has a chord, i.e., an edge connecting two vertices which are not adjacent in the cycle. Chordal bipartite graphs were discussed by GOLUMBIC [9] in relation with perfect Gaussian elimination for nonsymmetric matrices. Chordal bipartite graphs and totally-balanced matrices are equivalent in the following sense:

(3.8) Given a chordal bipartite graph $H = (\{1,2,\ldots,n\},\{1,2,\ldots,m\},E)$ define the $n\times m$ (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $(i,j) \in E$. Then A is totally-balanced.
Given an $n\times m$ totally-balanced matrix $A = (a_{ij})$ define the bipartite graph $H = (\{1,2,\ldots,n\},\{1,2,\ldots,m\},E)$ by $E = \{(i,j) \mid a_{ij} = 1\}$. Then H is a chordal bipartite graph.

An edge (i,j) of a bipartite graph is *bisimplicial* if the subgraph induced by all vertices adjacent to i and j is a complete bipartite graph. Let $M = (m_{ij})$ be a nonsingular nonsymmetric matrix. We can construct a bipartite graph from M equivalent to (3.8) where edges correspond to nonzero elements m_{ij}. If (i,j) is a bisimplicial edge in the bipartite graph, then using m_{ij} as a pivot in the matrix M to make m_{ij} to 1 and all other entries in the ith row and jth column equal to 0 does not change any zero element into a nonzero element. This is important since sparse matrices are represented in computers by its nonzero elements. GOLUMBIC [9] proved that a chordal bipartite graph has a bisimplicial edge. This result immediately follows from our result. Let i_0 be a nest row, i.e., $\sigma(i_0) = 1$ and let E_0 be such that for all columns E covering i_0 we have $E_0 \subseteq E$. Then the edge corresponding to E_0 and i_0 is a bisimplicial edge.

REFERENCES

[1] ANSTEE, R.P. & M. FARBER (1982), Characterizations of totally balanced matrices, Research Report CORR 82-5, Faculty of Mathematics, University of Waterloo.

[2] AHO, A.V., J.E. HOPCROFT & J.D. ULLMAN (1974), *The Design and Analysis of Computer Algorithms*, (Addison-Wesley, Reading, MA).

[3] BERGE, C. (1972), Balanced matrices, *Math. Programming* 2, 19-31.

[4] BROUWER, A.E. & A. KOLEN (1980), A super-balanced hypergraph has a nest point, Report ZW 146/80, Mathematisch Centrum, Amsterdam.

[5] FARBER, M. (1982), Characterization of Strongly Chordal Graphs, to appear in *Discrete Mathematics*.

[6] FARBER, M. (1982), Domination, Independent Domination and Duality in Strongly Chordal Graphs. Research Report CORR 82-2, Faculty of Mathematics, University of Waterloo.

[7] FULKERSON, D.R., A.J. HOFFMANN & R. OPPENHEIM (1974), On balanced matrices, *Math. Programming Study* 1, 120-132.

[8] GILES, R. (1978), A balanced hypergraph defined by certain subtrees of a tree, *Ars Combinatoria* 6, 179-183.

[9] GOLUMBIC, M.C. (1981), *Algorithmic Graph Theory and Perfect Graphs*, (Academic Press, New York).

[10] KOLEN, A., (1982), Location problems on trees and in the rectilinear plane, Ph.D. thesis, Mathematisch Centrum, Amsterdam.

[11] LOVÁSZ, L. (1979), *Combinatorial Problems and Exercises* (Akadémiai Kiadó, Budapest), 528.

[12] TAMIR, A. (1980), A class of balanced matrices arizing from location problems, Report Department of Statistics, Tel-Aviv University.