stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNE (DEPARTMENT OF OPERATIONS RESEARCH)

BW 147/81 SEPTEMBER

A.W.J. KOLEN

# A POLYNOMIAL-TIME ALGORITHM FOR SOLVING THE SET COVERING PROBLEM ON A TOTALLY-BALANCED MATRIX

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: 68E99, 05B40

# A polynomial-time algorithm for solving the set covering problem on a totally-balanced matrix

by

Antoon Kolen

ABSTRACT

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. Let A be an nxm totally-balanced matrix. We give an $O((\min\{m,n\})^2 \max\{m,n\})$ algorithm to solve the set covering problem defined on A; a tree location problem serves as an example of such a set covering problem. We also give an algorithm which recognizes an nxm totally-balanced matrix ($m \leq n$) in $O(nm^2)$ time.

KEY WORDS & PHRASES: *Set covering problem, balanced matrices, location theory*

## 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 [1] and FULKERSON et al. [3]. We consider a more restrictive class of matrices called totally-balanced (LOVÁSZ [7]). 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.

Let $A = (a_{ij})$ be an nxm totally-balanced matrix and let $c_j (j=1,2,\ldots,m)$ be nonnegative integers. The set covering problem is given by

$$(P) \qquad \min \sum_{j=1}^{m} c_j x_j$$

$$s.t \sum_{j=1}^{m} a_{ij} x_j \geq 1 , \quad i = 1,2, \ldots, n,$$

$$x_j \in \{0,1\} , \quad j = 1,2, \ldots, m.$$

The dual problem is given by

$$(D) \qquad \max \sum_{i=1}^{n} y_i$$

$$s.t \sum_{i=1}^{n} y_i a_{ij} \leq c_j , \quad j = 1,2, \ldots, m,$$

$$y_i \geq 0 \qquad , \quad i = 1,2, \ldots, n.$$

For an arbitrary (0,1)-matrix A the optimum value of (D) is less than or equal to the optimum value of (P). It is well known (FULKERSON et al. [3]) that in case of balanced matrices problems (P) and (D) when solved as linear programming problems give integral solutions. Due to the result of KHACHIAN [5] we know that both problems can be solved in polynomial time. For the case that A is totally-balanced we give an $O((\min\{m,n\})^2 \max\{m,n\})$ algorithm to solve both problems and give a constructive proof of strong duality.

As an example of problem (P) we consider the following problem.

EXAMPLE 1.1. Let $T = (V,E)$ be a tree with vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and edge set E. Each edge $e \in E$ has a positive length $\ell(e)$. The distance $d(v_i,v_j)$ between two vertices $v_i$ and $v_j$ is defined to be the length of the unique shortest path from $v_i$ to $v_j$. Let $J \subseteq \{1,2, \ldots, n\}$, $|J| = m$ and let $r_j$ $(j \in J)$ be nonnegative numbers. Define $T_j = \{v \in V \mid d(v,v_j) \leq r_j\}$ $(j \in J)$. Let $A = (a_{ij})$ be the nxm (0,1)-matrix defined by $a_{ij} = 1$ if and only if $v_i \in T_j$. It was first proved by GILES [4] that A is totally-balanced. We consider $v_j$ as the possible location of a facility, $T_j$ as the set of clients that can be served by $v_j$ (clients are located at vertices) and $c_j$ as the cost of establishing a facility at $v_j$ $(j \in J)$. The set covering problem (P) is the problem of finding facility locations which can serve all clients at minimum cost. This problem was solved in $O(n^2)$ time by the author [6]. In the same paper it was shown that totally-balanced matrices also occur in the simple plant location problem on the tree and an $O(n^3)$ algorithm to solve this problem was given. □

A (0,1)-matrix $A = (a_{ij})$ is in *standard form* if $a_{ik} = a_{i\ell} = a_{jk} = 1$ implies that $a_{j\ell} = 1$ for all $i,j,k,\ell$ with $i < j$ and $k < \ell$. Note that if a matrix A is in standard form its transpose $A^t$ is also in standard form. In Section 2 we show how to solve the set covering problem on a nxm matrix in standard form in $O(nm)$ time. In Section 3 we give an $O(nm^2)$ algorithm to transform an nxm totally-balanced matrix into a matrix in standard form. Due to the fact that a matrix is in standard form if its transpose is in standard form, this leads to an $O((\min\{m,n\})^2 \max\{m,n\})$ algorithm to solve the set covering problem on a totally-balanced matrix. An algorithm of the same complexity for recognizing a totally-balanced matrix is also given in Section 3.

## 2. THE SET COVERING ALGORITHM

In this section we solve the set covering problem (P), where we assume that the matrix $A = (a_{ij})$ is in standard form. We shall construct a dual feasible solution y and a primal feasible solution x such that if $I = \{i \mid y_i > 0\}$ and $J = \{j \mid x_j = 1\}$, then the following holds:

$$\sum_{j=1}^{n} y_i a_{ij} = c_j \text{ for all } j \in J, \tag{2.1}$$

and

(2.2) for each $i \in I$ there is at most one $j \in J$ such that $a_{ij} = 1$.

Note that since x is a primal feasible solution we can replace at most one by exactly one in (2.2). It follows from (2.1) and (2.2) that the values of the primal and dual feasible solutions are equal. Therefore both solutions are optimal.

The dual feasible solution is found by a greedy approach. The value of $y_i$ is determined according to increasing index and taken to be as large as possible. This procedure is formulated in the Dual algorithm.

Dual algorithm

for i := 1 step 1 to n

(2.3) $$\text{do } y_i := \min_{j:a_{ij}=1} \{c_j - \sum_{k=1}^{i-1} y_k a_{kj}\} \text{ od.}$$

EXAMPLE 2.1. The matrix and costs of the example as well as the result of the Dual algorithm are given in Figure 2.2.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | $y_1 = 2$ |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | $y_2 = 0$ |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | $y_3 = 0$ |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | $y_4 = 2$ |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | $y_5 = 0$ |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | $y_6 = 1$ |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | $y_7 = 1$ |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | $y_8 = 0$ |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | $y_9 = 0$ |
| 2 | 3 | 2 | 1 | 2 | 3 | 2 | cost |
| 0 | 1 | 2 | 1 | 2 | 3 | 2 | cost after subtracting $y_1$ |
| 0 | 1 | 0 | 1 | 2 | 1 | 2 | cost after subtracting $y_1+y_4$ |
| 0 | 1 | 0 | 0 | 2 | 1 | 1 | cost after subtracting $y_1+y_4+y_6$ |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | cost after subtracting $y_1+y_4+y_6+y_7$ |

Figure 2.2. Example of the Dual algorithm.

The Primal algorithm has as input the set $I = \{i \mid y_i > 0\}$ and the matrix A and as output a subset of columns J which will define a primal optimal solution. A row i is *covered by* column j in the matrix $A = (a_{ij})$ if $a_{ij} = 1$.

Primal algorithm.
Delete all columns which do not correspond to a tight constraint; $J := \emptyset$ ;
while there are still columns left
do add the last column to J;
delete all columns from the matrix that cover a row $i \in I$ which is also covered by the chosen last column
od.

EXAMPLE 2.3. Apply the Primal algorithm to Example 2.1. We have $I = \{1,4,6,7\}$ The Primal algorithm starts by deleting columns 5 and 6 which do not correspond to a tight constraint; $J := \emptyset$.

Iteration 1 : $J := \{7\}$; delete column 4 (columns 4 and 7 cover row $6 \in I$) and column 2(columns 2 and 7 cover row $7 \in I$).
Iteration 2 : $J := \{7,3\}$
Iteration 3 : $J := \{7,3,1\}$.

The value of the primal feasible solution defined by $x_j = 1$ if and only if $j \in J$ is equal to the value of the dual feasible solution, namely 6. □

With respect to the set I constructed by the Dual algorithm we call a column k a *blocking column* for row i if row i is covered by column k and for all rows j(j > i) which are covered by column k we have $j \notin I$. If we let j(i) denote the index of a constraint for which the minimum is attained in (2.3) during iteration i of the Dual algorithm, then we note that constraint j(i) is a tight constraint and column j(i) is a blocking column for row i.

THEOREM 2.4. *The* (0,1)*-solution defined by* $x_j = 1$ *if and only if* $j \in J$ *is a primal optimal solution.*

PROOF. It is clear that the dual feasible solution y and primal solution x satisfy (2.1) and (2.2). So in order to show that x is a primal optimal solution we have to show that it is a feasible solution, i.e., that the set of columns J covers all rows of the matrix. We shall prove this using

induction on the number of columns. The induction hypothesis is that all rows for which there is a blocking column are covered by the set of columns constructed by the Primal algorithm. Note that at the beginning of the while statement each row is covered by a blocking column. Let $\ell$ be the last column and delete all columns that cover a row $i \in I$ which is also covered by column $\ell$. We shall prove that for a row j which is not covered by column $\ell$ none of the deleted columns is a blocking column for row j. Then by the induction hypothesis this proves that J covers all rows. Suppose row j is covered by column k but not by column $\ell$ and columns k and $\ell$ both cover a row $i \in I$. If $j > i$, then since the matrix is in standard form this would imply that row j is covered by column $\ell$. Therefore $j < i$. Since $i \in I$ it follows that column k is not a blocking column for row j. □

## 3. THE STANDARD FORM TRANSFORMATION

In this section we give an $O(nm^2)$ algorithm which transforms an nxm totally-balanced matrix into a matrix in standard form as well as an $O(nm^2)$ algorithm which recognizes an nxm totally-balanced matrix.

Let $A = (a_{ij})$ be an nxm totally-balanced matrix. We consider column j ($j=1,2,\ldots,m$) 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 $\tau:\{E_1,E_2\ldots,E_m\} \to \{1,2\ldots,m\}$ corresponding to a transformation of the columns of A ($\tau(E_i) = j$ indicates 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. In 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 $G_r,\ldots,G_1$. 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 occur in 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 $\tau(E) \geq m-i+2$, we cannot distinguish between the rows b and c, i.e., $b \in E$ if and only if $c \in E$. At be beginning of iteration 1 all rows occur in the same group. Let $G_r,\ldots,G_1$ be the partitioning into groups at the beginning of iteration i ($1 \leq i \leq m$). For each column E not yet determined we calculate the vector $d_E$ of length r, where $d_E(j) = |G_{r-j+1} \cap E|$ $(j=1,2,\ldots,r)$. A column E for which $d_E$ is a lexicographically largest vector is the column determined in iteration i with $\tau(E) = m-i+1$. After we have determined E we can distinguish between some elements in the same group G if $1 \leq |G \cap E| < |G|$. If this is the case we shall take rows in $G \setminus E$ 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 $j = r,r-1,\ldots,1$ 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 $G_j = G_j \setminus E$. The algorithm ends after m iterations with a partitioning into groups, say $G_r,\ldots,G_1$. The permutation $\sigma$ 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 $\sum_{j=1}^{i-1} |G_j|+1,\ldots,\sum_{j=1}^{i} |G_j|$ 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 $O(nm^2)$.

EXAMPLE 3.1. The 9x7 (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 $\sigma$ is given by $\sigma$: $(6,7,8,1,2,4,9,3,5) \rightarrow (1,2,3,4,5,6,7,8,9)$. The mapping $\tau$ is given by $\tau$: $(E_7,E_6,E_1,E_3,E_5,E_2,E_4) \rightarrow (1,2,3,4,5,6,7)$. The transformed matrix is the one used in Example 2.1. □

A mapping $\sigma$: $\{1,2,\ldots,n\} \rightarrow \{1,2,\ldots,n\}$ is *a nest ordering with respect to* $E_1,\ldots,E_m$ 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 $\sigma(k) \geq i$ $(i = 1,2,\ldots,n)$. In a previous paper (BROUWER & KOLEN [2]) it was shown that there is a row of a totally-balanced matrix such that all columns covering this row can be totally ordered by inclusion. By inspection we can find this row in $O(nm^2)$ time. Give this row number 1 and delete the row from the matrix. Let $A_i$ be the matrix we get from A by deleting the rows with numbers $1,2,\ldots,i$. Then since $A_i$ is still totally-balanced there is a row with the property that all columns of $A_i$ covering this row can be totally ordered by inclusion. Give this row number i+1 $(i=1,2,\ldots,n-1)$. In this way we find a number for each row in $O(n^2m^2)$ time. Clearly the mapping defined above is a nest ordering.

We shall show that a mapping $\sigma$ produced by the transformation algorithm is a nest ordering with respect to $E_1,\ldots,E_m$. Since the algorithm takes $O(nm^2)$

time this a more efficient way of finding a nest ordering as well as a constructive proof of the fact that there is a row in a totally-balanced matrix with the property that all columns covering this row can be totally ordered by inclusion. By a *lexicographical 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 smaller than or equal to $E_j$ if $b_{E_i}$ is lexicographically smaller than or equal to $b_{E_j}$. Ties, which only occur when two subsets contain the same elements, are broken arbitrarily. Let $E_1, E_2$ be two columns. We call $E_1$ and $E_2$ *comparable* if $E_1 \subseteq E_2$ or $E_2 \subseteq E_1$. $E_1$ and $E_2$ are *incomparable* if they are not comparable.

LEMMA 3.2. *Let* A *be a matrix such that the ordering of the rows form a nest ordering with respect to the columns, and the columns are ordered in lexicographically increasing order. Then the matrix* A *is in standard form.*

PROOF. Suppose $a_{ik} = a_{i\ell} = a_{jk} = 1$, $i < j, k < \ell$. Since $i \in E_k \cap E_\ell$ it follows 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 smaller than or equal to $E_\ell$ it follows that $E_k^i \subseteq E_\ell^i$. Hence $a_{jk} = 1$ implies that $a_{j\ell} = 1$. □

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* $\sigma$ *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* $\sigma(i) < \sigma(j)$.

PROOF. Consider the iteration in 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 $\sigma(i) < \sigma(j)$. □

COROLLARY 3.4. *Let* $\sigma$ *and* $\tau$ *be the mappings constructed by the algorithm for a matrix* A. *First recorder the rows according to* $\sigma$. *Then* $\tau$ *is a lexicographic ordering of the columns.*

PROOF. Let $E_1, E_2$ be two columns such that $\tau(E_1) < \tau(E_2)$. If $E_1$ and $E_2$ are comparable, then $E_1 \subseteq E_2$. If $E_1$ and $E_2$ are incomparable, then it follows from Lemma 3.3. that the largest element in $E_2 \backslash E_1$ with respect to $\sigma$ is greater than any element in $E_1 \backslash E_2$ and hence $b_{E_1}$ is lexicographically smaller than $b_{E_2}$. □

Let $\sigma$ and $\tau$ be the mapping constructed by the algorithm applied on a totally balanced matrix A. If $\sigma$ is a nest ordering with respect to the columns of A, then it follows from Lemma 3.2 and Corollary 3.4 that if the matrix A is transformed according to $\sigma$ and $\tau$, then it is in standard form. We shall prove that the mapping $\sigma$ constructed by the algorithm applied on a totally-balanced matrix is a nest ordering with respect to the columns of the matrix using induction on the number of rows. If the number of rows is equal to 1, then the statement is true. Suppose the statement is true for all matrices with less than n rows, and let A be a nxm totally-balanced matrix given by its columns $E_1, \ldots, E_m$. Let $\sigma$ be the mapping constructed by the algorithm and let $i_0$ be the row of A such that $\sigma(i_0) = 1$. Define $A_1$ to be the matrix we get from A by deleting row $i_0$. Apply the algorithm on $A_1$. We shall prove in Lemma 3.5. that there exists a mapping $\sigma_1$ constructed by the algorithm applied on $A_1$ such that $\sigma_1(i) = \sigma(i)-1$ for all $i \neq i_0$. By the induction hypothesis it follows that $\sigma_1$ is a nest ordering with respect to the columns of $A_1$. In order to prove that $\sigma$ is a nest ordering with respect to the columns of A we have to show that all columns covering row $i_0$ can be totally ordered by inclusion. This will be proved in Theorem 3.7. After giving this outline of the correctness proof of the algorithm let us turn to the details.

LEMMA 3.5. *There exists a mapping* $\sigma_1$ *constructed by the algorithm applied on* $A_1$ *such that* $\sigma_1(i) = \sigma(i)-1$ *for all* $i \neq i_0$.

PROOF. It is sufficient to prove that at each iteration of the algorithm applied on $A_1$ we can choose the same column as at the corresponding iteration of the algorithm applied on A. Consider the partitioning into groups, say $G_r, \ldots, G_2, G_1$ at the beginning of iteration i of the algorithm applied on A and assume that in the algorithm applied on $A_1$ we have chosen the same column in the first i-1 iterations ($1 \leq i \leq m$) as in the corresponding iterations of the algorithm applied on A. If $|G_1| > 1$, then the partitioning

belonging to the algorithm applied on $A_1$ is given by $G_r,\ldots,G_2,G_1\backslash\{i_0\}$, else the partitioning is given by $G_r,\ldots,G_2$. Let $E_1$ be the column chosen in iteration i of the algorithm applied on A. In order to prove that we can also choose $E_1$ in iteration i of the algorithm applied an $A_1$ it is sufficient to prove $|G_1 \cap E_1| \geq |G_1 \cap E_2|$ implies that $|(G_1\backslash\{i_0\}) \cap E_1| \geq |(G_1\backslash\{i_0\}) \cap E_2|$ for all columns $E_2$ which have not yet been determined. Note that if $i_0 \in E_1$, then $G_1 \subseteq E_1$. If this was not the case, then after this iteration we would have $i_0 \in G_2$ which contradicts $\sigma(i_0) = 1$. If $i_0 \in E_1$ and $|G_1 \cap E_1| = |G_1 \cap E_2|$, then $G_1 \cap E_1 = G_1 \cap E_2 = G_1$ and therefore $|(G_1\backslash\{i_0\}) \cap E_1| = |(G_1\backslash\{i_0\}) \cap E_2|$. If $i_0 \in E_1$ and $|G_1 \cap E_1| \geq |G_1 \cap E_2| + 1$, then $|(G_1\backslash\{i_0\}) \cap E_1| = |G_1 \cap E_1| - 1 \geq |G_1 \cap E_2| \geq |(G_1\backslash\{i_0\}) \cap E_2|$.
If $i_0 \notin E_1$, then $|(G_1\backslash\{i_0\} \cap E_1)| = |G_1 \cap E_1| \geq |G_1 \cap E_2| \geq |(G_1\backslash\{i_0\}) \cap E_2|$ □

According to the previous outline of the proof we have to show that all columns of A covering row $i_0$ can be totally ordered by inclusion. Suppose that there are two incomparable columns $E_1$ and $E_2$ covering row $i_0$. Without loss of generality assume $\tau(E_1) < \tau(E_2)$. Let $i_1$ be the largest element with respect to $\sigma$ in $E_1\backslash E_2$, and let $i_2$ be the largest element with respect to $\sigma$ in $E_2\backslash E_1$. 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 generalise the definition of an 2-chain to an m-chain using the following definition. A column $E_1$ *separates* i from j if $i \in E_1$, $j \notin E_1$ and for all columns $E_2$ with $\tau(E_2) > \tau(E_1)$ we have $i \in E_2$ if and only if $j \in E_2$. 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.
We call $(i_0,i_1,i_2,\ldots,i_m,E_1,E_2,\ldots,E_m)$ an *m-chain* $(m \geq 2)$ if

1. $i_j \in E_k \iff j = k$ 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 $\sigma$ in $E_j$ which is not contained in $E_{j-1}$ $(j=1,2,\ldots,m)$, where $E_0 = E_2$.

THEOREM 3.6. *An m-chain can be extended to an* m+1*-chain* $(m \geq 2)$.

PROOF. Since $\sigma(i_{m-1}) > \sigma(i_{m-2})$ and $i_{m-2}$ and $i_{m-1}$ are not covered by the same columns ($i_{m-1} \notin E_m$) it follows that there is 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 $\tau(E_{m+1}) > \tau(E_m)$ it follows that $i_{m-3} \notin E_{m+1}$. Repeating this argument for $E_{m-1},\ldots,E_3$ respectively shows that $i_0,\ldots,i_{m-2} \notin E_{m+1}$. 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 $i_{m-1} \notin E_m$ it follows that $E_m$ and $E_{m+1}$ are incomparable. Let $i_{m+1}$ be the largest row with respect to $\sigma$ in $E_{m+1}$ which is not contained in $E_m$. 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 an m+1-chain we have to prove that $i_{m+1} \notin E_k$ for $k = 1,\ldots,m$. We already saw that $i_{m+1} \notin E_m$. Suppose $i_{m+1} \in E_k$ for some $k(1 \leq k \leq m-1)$. 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 $\sigma$ in $E_k$ which is not contained in $E_{k-1}$ and $\sigma(i_{m+1}) > \sigma(i_k)$ it follows that $i_{m+1} \in E_{k-1}$. 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 since $i_{k-1} \in E_{k-1} \cap E_{k+1}$ $E_{k-1}$ and $E_{k+1}$ are comparable with respect to all rows i with $\sigma(i) \geq \sigma(i_{k-1}) > 1$. We conclude that $i_{m+1} \notin E_k$ for all $k = 1,\ldots,m$. □

THEOREM 3.7. *All columns covering row* $i_0$ *can be totally ordered by inclusion.*

PROOF. If there are two incomparable columns covering row $i_0$, then there exists an 2-chain. It follows from Theorem 3.6 that we can extend this chain infinitely many times. This contradicts the fact that the number of rows of A is finite. □

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

THEOREM 3.8. *A* (0,1)*-matrix* A *is totally-balanced if and only if the mapping* $\sigma$ *constructed by the transformation algorithm applied on* A *is a nest ordering.*

PROOF. If A is totally-balanced, then we proved that $\sigma$ 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 all its row and columns sums equal to two. Let row $i_1$ be the smallest row with respect to $\sigma$ 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 $i_2 \in E_j \backslash E_k$ and $i_3 \in E_k \backslash E_j$, i.e. $E_j$ and $E_k$ are not comparable with respect to all rows i with $\sigma(i) \geq \sigma(i_1)$. Hence $\sigma$ is not a nest ordering. □

We can find $\sigma$ in $O(nm^2)$ time. Checking whether $\sigma$ 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 in any other iteration k with $k > i$. So we only have to check each pair of columns at most ones. This can be done in $O(nm^2)$ time. Hence the recognition also requires $O(nm^2)$ time.

REFERENCES


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

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

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

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

[5] KHACHIAN, L.G., *A polynomial algorithm in linear programming*, Doklady Akademie Nauk USSR 5 (1979) 244.

[6] KOLEN, A., *Minimum cost tree location problems*, Annals of Discrete Mathematics, to appear.

[7] LOVÁSZ, L., *Combinatorial problems and exercises*, Akadémiai Kiadó, Budapest (1979) 528.