stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 163/82 JUNI
(DEPARTMENT OF OPERATIONS RESEARCH)

A.W.J. KOLEN

SOLVING COVERING PROBLEMS AND THE UNCAPACITATED PLANT LOCATION PROBLEM ON TREES

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

1980 Mathematics subject classification: 90C05, 90C10, 90B20

Solving covering problems and the uncapacitated plant location problem on trees *)

by

A.W.J. Kolen**)

ABSTRACT

Given a tree network on n vertices, a neighborhood subtree is defined as the set of all points on the tree within a certain radius of a given point, called the center. It is shown that for any two neighborhood subtrees containing the same endpoint of a longest path in the tree one is contained in the other. This result is then used to obtain $O(n^2)$ algorithms for the minimum cost covering problem and the minimum cost operating problem as well as an $O(n^3)$ algorithm for the uncapacitated plant location problem on the tree.

KEY WORDS & PHRASES: *location theory, minimum cost covering problem, minimum cost operating problem, uncapacitated plant location problem, integer programming*

*) This report will be submitted for publication elsewhere.

**) EUR Erasmus Universiteit Rotterdam
Burgemeester Oudlaan 50, 3062 PA Rotterdam

1. INTRODUCTION

Consider the problem of a company which has to provide goods to a prespecified set of clients. In order to execute this task the company has to establish a number of facilities. For each possible facility location it is known what the cost is of establishing a facility at that location and which clients can be served from that location. It is assumed that clients are located on a network of roads. In the *minimum cost covering problem* each facility can serve only clients within a given distance from that facility and the problem is to serve all clients at minimum cost. In the *minimum cost operating problem* each facility also can serve only clients within a given distance from that facility but it is not necessary to serve all clients. If a client is not served there is a penalty cost involved. This penalty cost may represent for example a loss of future orders for the company. The *minimum cost operating problem* is to minimize the sum of the cost of establishing facilities and of the penalty cost of not serving clients. In the *uncapacitated plant location problem* each facility can serve all clients but beside the cost of establishing facilities there is a cost for transporting goods from a facility to a client which is assumed to be linear with respect to the distance traveled. The *uncapacitated plant location problem* is to minimize the sum of the cost of establishing facilities and of the transportation cost. The solution of these three problems is the subject of this paper in which we will assume that the underlying network contains no cycles, i.e., is a tree network.

Consider the linear programming problem given by

$$(1.1)\qquad \begin{aligned} &\min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} d_i z_i \\ &\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j + z_i \ge b_i, && i = 1,2,\dots,n, \\ &\qquad\qquad\qquad x_j \ge 0\ , && j = 1,2,\dots,m, \\ &\qquad\qquad\qquad z_i \ge 0\ , && i = 1,2,\dots,n, \end{aligned}$$

where $A = (a_{ij})$ is an $n \times m(0,1)$-matrix, $c_j \ge 0$, $j = 1,2,\dots,m$, $d_i \ge 0$, $i = 1,2,\dots,n$, $b_1 \ge b_2 \ge \dots \ge b_n \ge 0$. The corresponding dual is given by

$$(1.2)\qquad \max \sum_{i=1}^{n} b_i y_i$$
$$\text{s.t.} \sum_{i=1}^{n} y_i a_{ij} \le c_j, \quad j = 1,2,\dots,m,$$
$$0 \le y_i \le d_i, \quad i = 1,2,\dots,n.$$

Let us define the 2×2 matrix A^* which plays an important role in this paper by

$$A^* = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

If the matrix A has the property that it does not contain A^* as a submatrix, then HOFFMAN, KOLEN & SAKAROVITCH [5] give an O(nm) algorithm to solve both problem (1.1) and (1.2). If b_i is integer, $i = 1,2,\dots,n$, then the algorithm constructs an integer optimal solution to problem (1.1).

We will show that the three tree location problems considered can be formulated as

$$(1.3)\qquad \min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} d_i z_i$$
$$\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j + z_i \ge 1 \quad , i = 1,2,\dots,n,$$
$$x_j \in \{0,1\},\ j = 1,2,\dots,m,$$
$$z_i \in \{0,1\},\ i = 1,2,\dots,n.$$

Furthermore we show that the (0,1)-matrix $A = (a_{ij})$ of (1.3) can be transformed by row and column permutations into a matrix which does not contain A^* as a submatrix. We can then use the algorithm of HOFFMANN, KOLEN & SAKAROVITCH [5] to solve the LP-relaxation of (1.3) since the algorithm garantees a (0,1) optimal solution. By using this approach we reverse history. Originally the solution procedure developed for the minimum cost covering problem as presented at ISOLDE II motivated the research for the more general problem given by (1.1).

In Section 2 we formally describe the three tree location problems we consider in this paper and show that they can be considered as a special case of problem (1.3).

In section 3 we show that the (0,1)-matrix $A = (a_{ij})$ of the location problems formulated as a special case of problem (1.3) can be transformed by row and column permutations into a matrix which does not contain A^* as a submatrix. We complete this section by describing the algorithm of HOFFMAN, KOLEN & SAKAROVITCH [5] which is used to solve the location problems.

In Section 4 we give an example of the solution procedure for each of the three tree location problems.

In Section 5 we define the dual problems of the location problems and give an economic interpretation of the dual problems.

2. PROBLEM FORMULATION

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 $\ell(e)$. One can think of T as a planar road network. A *point* on T can be a vertex or a point anywhere along an edge. The *distance* $d(x,y)$ between two points x and y on T is defined to be the length of the shortest path between x and y. The shortest path between x and y is denoted by $P(x,y)$. A *neighborhood subtree* is defined as the set of all points on T within a certain distance (called the *radius*) from a given point (called the *center*). Let s_j, $j = 1,2,\dots,m$ be possible locations for a facility at the tree T. Assume that a facility located at s_j can serve only clients which are located within distance r_j, $r_j \geq 0$ from s_j. The cost of establishing a facility at s_j is given by c_j, $c_j > 0$. We assume clients to be located at vertices of the tree and also assume that the the facility locations belong to the vertex set. This can be done without loss of generality since if this is not the case we simply can add the corresponding point to the vertex set and adjust the edge set accordingly. We say that facility j is *open* if there is a facility established at s_j; otherwise it is said to be *closed*. The *minimum cost covering problem* is to serve all clients at minimum cost. If we define the $n\times m(0,1)$-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $d(v_i,s_j) \leq r_j$, i.e., the client located at v_i can be served by a facility located at s_j, then the *minimum cost covering problem* can be formulated as

$$(2.1) \qquad \min \sum_{j=1}^{m} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1,2,\dots,n \qquad (2.2)$$

$$x_j \in \{0,1\}, \; j = 1,2,\dots,m,$$

where $x_j = 1$ if and only of facility j is open. The constraints (2.2) ensure that each client is served by at least one open facility. Note that problem (2.1) is a special case of (1.3) if we take $d_i = \infty$, $i = 1,2,\ldots,n$.

Let us relax the condition that each client has to be served by at least one open facility and assume that there is a penalty cost d_i if a client located at v_i is not served by a facility, $i = 1,2,\ldots,n$. The *minimum cost operating problem* is to minimize the sum of the cost of establishing facilities and of the penalty cost of not serving clients. This problem can be formulated as

$$
\begin{aligned}
(2.3)\qquad & \min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} z_i d_i \\
& \text{s.t.} \sum_{j=1}^{m} a_{ij} x_j + z_i \geq 1 \qquad i = 1,2,\ldots,n \qquad (2.4) \\
& \qquad x_j \in \{0,1\},\ j = 1,2,\ldots,m, \\
& \qquad z_i \in \{0,1\},\ i = 1,2,\ldots,n.
\end{aligned}
$$

The constraints (2.4) ensure that $z_i = 1$ if and only if $\sum_{j=1}^{m} a_{ij}x_j = 0$, i.e., the client located at v_i is not served by an open facility. Note that problem (2.3) is a special case of (1.3).

For the next problem let us assume that each facility can serve all clients and that there is a transportation cost associated with transporting goods from a facility to a client which is linear with respect to the traveled distance. The *uncapacitated plant location problem* is to minimize the sum of the cost of establishing the facilities and of the transportation cost. This problem can be formulated as

$$
\begin{aligned}
(2.5)\qquad & \min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{m} w_i d(v_i, s_j) f_{ij} \\
& \text{s.t.} \sum_{j=1}^{m} f_{ij} = 1, \qquad i = 1,2,\ldots,n, \qquad (2.6) \\
& \qquad x_j - f_{ij} \geq 0, \qquad i = 1,2,\ldots,n,\ j = 1,2,\ldots,m, \quad (2.7) \\
& \qquad f_{ij} \geq 0, \qquad i = 1,2,\ldots,n,\ j = 1,2,\ldots,m, \\
& \qquad x_j \in \{0,1\},\ j = 1,2,\ldots,m,
\end{aligned}
$$

where f_{ij} is the fraction of the demand of the client located at v_i which is supplied by the facility at s_j. The constraints (2.6) ensure that each client is served The constraints (2.7) ensure that each client is served only by open facilities. Formulation (2.5) with $w_i d(v_i,s_j)$ replaced by arbitrary transportation cost t_{ij} is the standard formulation of the uncapacitated plant location problem in the literature. For a survey on this problem see KRARUP & PRUZAN [7]. An algorithm for the uncapacitated plant location problem which performs well in practice is given by ERLENKOTTER [3]. The basic idea of this algorithm can already be found in BILDE & KRARUP [2].

The general uncapacitated plant location problem (also often called the *simple plant location problem*) is NP-hard. We will develop a polynomial-time algorithm for the problem defined on a tree as is given by (2.5). By combining constraints (2.6) and (2.7) it follows that at least one facility must be open. Given a nonempty set of open facilities, defined by the (0,1)-vector x, the best value v(x) corresponding to this set of open facilities is obtained by supplying each client from the closest open facility. Therefore we have

$$(2.8) \qquad v(x) = \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} w_i \min_{j:x_j=1}\{d(v_i,s_j)\}.$$

In order to describe our algorithm we have to formulate the uncapacitated problem as a special case of (1.3). This is possible due to a reformulation of the problem which was mentioned to the author by TAMIR [10]. For each vertex v_i we calculate the set of distances to all possible facility locations s_j. If v_i is not a facility location itself, then we add zero to this set of distances. The elements of this set are ordered in increasing order, say, $0 = r_{i1} < r_{i2} < \ldots < r_{it(i)}$. Define $r_{i,t(i)+1} = \infty$. Note that t(i) may be strictly less than m since there may be two facilities at equal distance from v_i. Define the (0,1)-variable z_i^k, $i = 1,2,\ldots,n$, $k = 1,2,\ldots,t(i)$ by

$$z_i^k = \begin{cases} 1 \text{ if } v_i \text{ is not served by an open facility within distance } r_{ik} \text{ from } v_i, \\ 0 \text{ otherwise} \end{cases}$$

and define a_{ij}^k, $i = 1,2,\ldots,n$, $k = 1,2,\ldots,t(i)$, $j = 1,2,\ldots,m$ by

$$a_{ij}^k = \begin{cases} 1 \text{ if } d(v_i,s_j) \le r_{ik}, \\ 0 \text{ otherwise.} \end{cases}$$

Then the uncapacitated plant location problem can be formulated as

$$\text{(2.9)} \qquad \min \sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} \sum_{k=1}^{t(i)} w_i(r_{i,k+1} - r_{ik}) z_i^k$$

$$\text{s.t.} \sum_{j=1}^{m} a_{ij}^k x_j + z_i^k \geq 1, \qquad i = 1,2,\ldots,n,\ k = 1,2,\ldots,t(i), \quad (2.10)$$

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

$$z_i^k \in \{0,1\},\ i = 1,2,\ldots,n,\ k = 1,2,\ldots,t(i).$$

The constraints (2.10) ensure that $z_i^k = 1$ if and only if $\sum_{j=1}^m a_{ij}^k x_j = 0$, i.e., there is no open facility within distance r_{ik} from v_i. Note that problem (2.9) is a special case of problem (1.3). Since $w_i(r_{i,t(i)+1} - r_{i,t(i)}) = \infty$ we have $z_i^{t(i)} = 0$ in an optimal solution of (2.9) and hence $\sum_{j=1}^m a_{ij}^{t(i)} x_j \geq 1$, i.e., there is at least one open facility. (note that $a_{ij}^{t(i)} = 1$.) In order to show that formulations (2.5) and (2.9) of the uncapacitated plant location problem are equivalent we demonstrate that the best value $v(x)$ corresponding to a nonzero (0,1)-solution x of (2.9) is equal to (2.8). Let $r_{i\ell}$ be the shortest distance from v_i to an open facility, i.e.,
$r_{i\ell} = \min_{j\,:\,x_j=1}\{d(v_i,s_j)\}$. Then we have

$$\sum_{j=1}^{m} a_{ij}^k x_j \geq 1 \text{ for all } k,\ k \geq \ell$$

and

$$\sum_{j=1}^{m} a_{ij}^k x_j = 0 \text{ for all } k,\ k < \ell.$$

Therefore a minimal solution is given by $z_i^k = 0$ for all k, $k \geq \ell$ and $z_i^k = 1$ for all k, $k < \ell$. If $\ell = 1$, then $\sum_{k=1}^{t(i)} w_i(r_{i,k+1} - r_{ik}) z_i^k = 0 = w_i r_{i\ell}$ (note that $r_{i1} = 0$).
If $\ell > 1$, then $\sum_{k=1}^{t(i)} w_i(r_{i,k+1} - r_{ik}) z_i^k = \sum_{k=1}^{\ell-1} w_i(r_{i,k+1} - r_{ik}) = w_i(r_{i\ell} - r_{i1})$ $= w_i r_{i\ell}$. We conclude that $v(x)$ is given by (2.8).

3. ALGORITHM

In this section we show that the (0,1)-matrix $A = (a_{ij})$ of the location problems formulated as a special case of (1.3) can be transformed by row and column permutations into a matrix which does not contain A^* as a submatrix. We can then use the algorithm of HOFFMAN, KOLEN & SAKAROVITCH [5] given at the end of this section to solve the location problems.

Let $T = (\{1,2,\dots,n\},E)$ be a tree. We say that T is *rooted* if $d_{1n} \geq d_{2n} \geq \dots \geq d_{n-1n} \geq d_{nn}$, where d_{ij} denotes the distance from vertex i to vertex j. A tree $T = (V,E)$ on n vertices can be transformed into a rooted tree by a 1-1 mapping $\sigma : V \to \{1,2,\dots,n\}$ defined as follows. Choose a vertex v and calculate the distances from v to all other vertices and order them in nonincreasing order, say, $d(t_1,v) \geq d(t_2,v) \geq \dots \geq d(t_n,v)$, where $t_n = v$. Define $\sigma(t_i) = i$ $(i = 1,2,\dots,n)$.

EXAMPLE 3.1. Consider the tree given by Figure 3.2, where the length of an edge is indicated between parentheses.

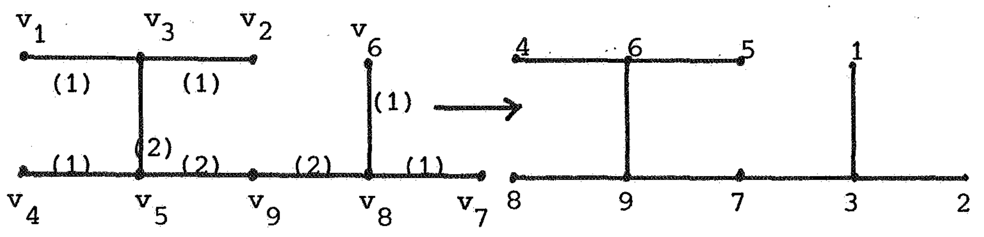

Figure 3.2. Tree of example and corresponding rooted tree

A 1-1 mapping $\sigma : V \to \{1,2,\dots,9\}$ which turns the tree into a rooted tree is given by $\sigma : (v_6,v_7,v_8,v_1,v_2,v_3,v_9,v_4,v_5) \to (1,2,3,4,5,6,7,8,9)$. □

In the minimum cost covering problem and minimum cost operating problem a column j of the (0,1)-matrix involved corresponds to the vertices which are contained in the neighborhood subtree $\{y \in T \mid d(y,s_j) \leq r_j\}$. In the uncapacitated plant location problem a row i,k of the (0,1)-matrix involved corresponds to the facility locations which are contained in the neighborhood subtree $\{y \in T \mid d(v_i,y) \leq r_{ik}\}$. If the tree is rooted, then rows in the uncapacitated plant location problem and columns in the other two problems correspond to subsets of $\{1,2,\dots,n\}$. We will order these subsets in a lexicographically nondecreasing order. A *lexicographic ordering* is defined as follows. Let E_1,E_2 be two subsets of $\{1,2,\dots,n\}$. If $E_1 \subseteq E_2$, then E_1 is lexicographic less than or equal to E_2. If $E_1 \not\subseteq E_2$ and $E_2 \not\subseteq E_1$, then E_1 is lexicographic less than E_2 if the largest element in $E_1 \backslash E_2$ is smaller than the largest element in $E_2 \backslash E_1$. Let $E_1,E_2,\dots,E_m$ be subsets of $\{1,2,\dots,n\}$. We say that the 1-1 mapping $\tau : \{1,2,\dots,m\} \to \{E_1,E_2,\dots,E_m\}$ is a *lexicographically nondecreasing ordering* if for all i,j, $i < j$ $\tau(i)$ is lexicographic less than or equal to $\tau(j)$. A lexicographically nondecreasing ordering can be obtained in $O(n\,m)$ time using a radix sort procedure (see AHO, HOPCROFT & ULLMAN [1]). When we mention a lexicographically nondecreasing ordering of neighborhood subtrees we mean a lexicographically nondecreasing ordering of the set of vertices contained in these neighborhood subtrees.

EXAMPLE 3.3. Consider the subsets of $\{1,2,\dots,9\}$ given by $E_1 = \{4,5,6,9\}$, $E_2 = \{4,5,6\}$, $E_3 = \{8,9\}$, $E_4 = \{6,7,8,9\}$, $E_5 = \{3,7,9\}$, $E_6 = \{1,2,3,7\}$ and $E_7 = \{1,2,3\}$. A lexicographically nondecreasing ordering τ is given by $\tau : (1,2,3,4,5,6,7) \to (E_7,E_2,E_6,E_1,E_5,E_3,E_4)$. For example $\tau(1)$ precedes $\tau(3)$ because $E_7 \subseteq E_6$, $\tau(5)$ precedes $\tau(6)$ because $7 < 8$, where 7 is the largest element in $E_5 \backslash E_3$ and 8 is the largest element in $E_3 \backslash E_5$. □

Let $T = (\{1,2,\dots,n\},E)$ be a rooted tree. It follows from Property 3 of the Appendix that vertex 1 is endpoint of a longest path in the tree T_1, where $T_1 = T$. Note that vertex 1 is a tipvertex of T_1, i.e., a vertex which is contained in exactly one edge of T_1 (see Figure 3.2). If we delete vertex 1 and the unique edge containing 1 from T_1, then the remaining graph is still a tree, called T_2. In general T_i is the subtree of T obtained by deleting the vertices $1,2,\dots,i-1$ and the edges containing at least one of those vertices from T, or equivalently, T_i is the subtree defined by the vertex set $\{i,i+1,\dots,n\}$ and all edges containing two of those vertices ($i = 1,2,\dots,n$). It follows from Property 3 of the Appendix that vertex i is endpoint of a longest path in T_i.

LEMMA 3.4. *Let* E *be a neighborhood subtree. If* $E \cap T_i \neq \emptyset$, *then* $E \cap T_i$ *is a neighborhood subtree of* T_i ($i = 1,2,\dots,n$).

PROOF. We have to prove that there exists a point $x_i \in T_i$ and a nonnegative number r_i such that $E \cap T_i = \{y \in T_i \mid d(y,x_i) \le r_i\}$. For $i = 1$ this is true since $E \cap T_1 = E$. Suppose $E \cap T_i = \{y \in T_i \mid d(y,x_i) \le r_i\}$ for some $x_i \in T_i$, $r_i \ge 0$. Assume $E \cap T_{i+1} \ne \emptyset$. Since i is a tipvertex of T_i there is a unique vertex $a(i) \in T_i$ adjacent to i. Define x_{i+1} and r_{i+1} as follows. If x_i lies on the edge $[i,a(i)]$, then $x_{i+1} = a(i)$ and $r_{i+1} = r_i - d(x_i,a(i))$, else $x_{i+1} = x_i$ and $r_{i+1} = r_i$. Then we have $E \cap T_{i+1} = \{y \in T_{i+1} \mid d(y,x_{i+1}) \le r_{i+1}\}$. □

In the Appendix we shall prove the following theorem.

<u>THEOREM 3.5</u>. *If* E_1 *and* E_2 *are two neighborhood subtrees containing the same endpoint of a longest path in the tree, then* $E_1 \subseteq E_2$ *or* $E_2 \subseteq E_1$.

This can be viewed as a generalization of a property of intervals. Let $[a,b]$ be a given interval. Let $x_1,\dots,x_m$ be points in the given interval and let $r_1,\dots,r_m$ be nonnegative numbers. Define the intervals $I_i = \{y \mid a \le y \le b, |y-x_i| \le r_i\}$, $i = 1,2,\dots,m$. If I_1 and I_2 are intervals containing a, then $I_1 \subseteq I_2$ or $I_2 \subseteq I_1$. A tree and a neighborhood subtree can be viewed as generalizations of an interval and a subinterval respectively. Theorem 3.5 expresses a generalization of the intersection property of intervals containing a given endpoint of the interval. The importance of this property is expressed in the following lemma.

<u>LEMMA 3.6</u>. *Let* $T = (\{1,2,\dots,n\},E)$ *be a rooted tree and let* $E_1,\dots,E_m$ *be neighborhood subtrees. Define the* $n\times m(0,1)$*-matrix* $A = (a_{ij})$ *by* $a_{ij} = 1$ *if and only if* $i \in E_j$. *If the columns of* A *are ordered in a lexicographically nondecreasing order, then the transformed matrix does not contain* A^* *as a submatrix.*

<u>PROOF</u>. Suppose A^* is a submatrix of the transformed matrix. Let the rows be $i,j,i < j$ and columns corresponding to E_k and E_ℓ, where E_k precedes E_ℓ in the lexicographic ordering. Since i is endpoint of a longest path in T_i we have according to Theorem 3.5 $(E_k \cap T_i) \subseteq (E_\ell \cap T_i)$ or $(E_\ell \cap T_i) \subseteq (E_k \cap T_i)$ (Note that according to Lemma 3.4 $E_k \cap T_i$ and $E_\ell \cap T_i$ are neighborhood subtrees). Since $j \in (E_k \cap T_i)\setminus(E_\ell \cap T_i)$ we have $(E_\ell \cap T_i) \subset (E_k \cap T_i)$. This implies however that in the lexicographic ordering E_ℓ precedes E_k, which contradicts our starting assumption. Hence the transformed matrix does not contain A^* as a submatrix. □

<u>COROLLARY 3.7</u>. *Let* $T = (\{1,2,\dots,n\},E)$ *be a rooted tree and let* $E_1,\dots,E_m$ *be neighborhood subtrees. Define the* $m\times n$ $(0,1)$*-matrix* $A = (a_{ij})$ *by* $a_{ij} = 1$ *if and only if* $j \in E_i$. *If the rows of* A *are ordered in a lexicographically nondecreasing order, then the transformed matrix does not contain* A^* *as a submatrix.*

<u>PROOF</u>. It follows from Lemma 3.6 that the transpose of the transformed matrix does not contain A^* as a submatrix. Since A^* is symmetric also the transformed matrix does not contain A^* as a submatrix. □

Lemma 3.6 indicates how the (0,1)-matrix involved in the formulation of the minimum cost covering and operating problem can be transformed such that the transformed matrix does not contain A^* as a submatrix. First we turn the tree into a rooted tree (this corresponds to a permutation of the rows) and then we order the columns in a lexicographically nondecreasing order.

Corollary 3.7 indicates how the (0,1)-matrix involved in the formulation of the uncapacitated plant location problem can be transformed to exclude A^* as a submatrix. First we turn the tree into a rooted tree (this corresponds to a permutation of the columns) and then we order the rows in a lexicographically nondecreasing order.

After we have transformed the matrix of problem (1.3) to exclude A^* as a submatrix we can use the algorithm of HOFFMAN, KOLEN & SAKAROVITCH [5] stated below. The correctness proof of the algorithm can be found in [5] and will not be repeated here. It suffices to say that the proof is based on the fact that the dual and primal solutions obtained are feasible and satisfy the complementary slackness relations of linear programming. In the algorithm we use the convention that <u>od</u> and <u>fi</u> indicate the end of a <u>do</u> and <u>if</u> statement respectively. A column j is said to *cover* row i if $a_{ij} = 1$. In the algorithm J denotes a set of indices j for which constant j is *tight*, i.e. $\sum_{i=1}^{n} y_i a_{ij} = c_j$ for the dual solution y constructed, I will be the index set of positive dual variables which make at least one constraint tight during the iteration in which that value was determined, and O will be the set of open facilities. As we saw in Section 2 the set of open facilities completely determines the optimal solution of problem (1.3)

ALGORITHM 3.8

```
begin  J:=∅;O:=∅;I:=∅;ĉ:=c;
       for i:=1 step 1 to n
       do $y_i$:=min{$d_i$,$\min_{j:a_{ij}=1} \hat{c}_j$};
          if $y_i > 0$ then if $y_i = \hat{c}_j$ for some j then choose the largest j and
                                let J:=J ∪ {j};I:=I ∪ {i}
                            fi;
                            $\hat{c}_j := \hat{c}_j - y_i$ for all j such that $a_{ij} = 1$
          fi
       od;
       while J≠∅
       do let k be the last column of J;
          O:=O ∪ {k};
          delete from J all columns which cover a row of I which is also
          covered by column k
       od
end
```

Note that the dual solution y is obtained by a greedy approach, i.e., y_i is determined by increasing index i and taken to be as large as possible with respect to the dual constraints. Examples of the solution procedures for each of the three tree location problems are given in the next section.

Let a (0,1)-matrix be *totally balanced* if it does not contain a square submatrix of size at least three with no identical columns and row and column sums equal to two. Figure 3.9 gives two examples of the type of submatrices which are forbidden in a totally balanced matrix

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} , \quad \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.8. Examples of forbidden submatrices

It is a trivial observation that if a (0,1)-matrix A does not contain A^* as a submatrix, then A is totally-balanced. Let $T = (\{v_1,\dots,v_n\},E)$ be a tree and let $E_1,\dots,E_m$ be neighborhood subtrees. Define the n×m (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $v_i \in E_j$. Since the property of being totally-balanced is not changed if we reorder the rows and columns of the matrix we have according to Lemma 3.6 that the matrix A is totally-balanced. This result was known before and first proved by GILES [4]. TAMIR [8] proved the following generalization of this result; Let $E_1,\dots,E_m,R_1,\dots,R_n$ be neighborhood subtrees. Define the n×m (0,1)-matrix $B = (b_{ij})$ by $b_{ij} = 1$ if and only if $R_i \cap E_j \neq \emptyset$. Then B is totally-balanced. The fact that this special totally-balanced matrix arising in the context of tree location theory can be transformed into a matrix which does not contain A^* as a submatrix makes one wonder whether this is true for arbitrary totally-balanced matrices. If the answer is yes, then again we can solve a whole class of integer programming problems defined on a totally-balanced matrix using the algorithm of HOFFMAN, KOLEN & SAKAROVITCH [5]. It is indeed possible to transform a n×m totally-balanced matrix into the desired form by an $O(m^2n)$ algorithm presented in [5].

4. EXAMPLES

In this section we demonstrate the solution procedure by giving an example for each of the three tree location problems.

4.1. The minimum cost covering problem

ALGORITHM 4.1.1.

```
begin make of the tree a rooted tree;
      construct a lexicographically nondecreasing ordering $\tau$ of the neighbor-
      hood subtrees;
      define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $i \in \tau(j)$;
      define $d_i = \infty$, $i = 1,2,\dots,n$;
      use Algorithm 3.8
end
```

EXAMPLE 4.1.2. Consider the tree T of Example 3.1 and let the neighborhood subtrees and their cost be given by Figure 4.1.3.

T_j	center	radius	cost
T_1	v_3	2	9
T_2	v_1	2	7
T_3	v_4	1	5
T_4	v_5	2	3
T_5	v_9	2	3
T_6	v_8	2	5
T_7	v_6	2	3

Figure 4.1.3. Data of the example

We turn T into a rooted tree as indicated by Example 3.1. Then the vertices contained in the neighborhood subtree T_j correspond to the set E_j given in Example 3.3, $j = 1,2,\ldots,7$. A lexicographically nondecreasing ordering τ is given by Example 3.3. The matrix A corresponding to the example and the successive iterations to obtain the dual solution y are

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\hat{c}_1 = 3,\ \hat{c}_2 = 7,\ \hat{c}_3 = 5,\ \hat{c}_4 = 9,\ \hat{c}_5 = 3,\ \hat{c}_6 = 5,\ \hat{c}_7 = 3.$

$y_1 = 3,\ I = \{1\},\ J = \{1\},\ \hat{c}_1 = 0,\ \hat{c}_3 = 2.$

$y_2 = 0.$

$y_3 = 0.$

$y_4 = 7,\ I = \{1,4\},\ J = \{1,2\},\ \hat{c}_2 = 0,\ \hat{c}_4 = 2.$

$y_5 = 0.$

$y_6 = 0.$

$y_7 = 2,\ I = \{1,4,7\},\ J = \{1,2,3\},\ \hat{c}_3 = 0,\ \hat{c}_5 = 1,\ \hat{c}_7 = 1.$

$y_8 = 1,\ I = \{1,4,7,8\},\ J = \{1,2,3,7\},\ \hat{c}_6 = 4,\ \hat{c}_7 = 0.$

$y_9 = 0.$

The total value of the solution y is 13. The while statement of Algorithm 3.8 operates on the submatrix of A with rows belonging to I and columns belonging to J.

I\J	1	2	3	7
1	1	0	1	0
4	0	1	0	0
7	0	0	1	1
8	0	0	0	1

Iteration 1: $0 = \{7\}$, column 3 is deleted since it covers row 7.

Iteration 2: $0 = \{2,7\}$.

Iteration 3: $0 = \{1,2,7\}$.

It is easy to check that facilities 1, 2, and 7 can serve all clients at a cost of 13. □

Making T a rooted tree takes $O(n \log n)$time. The lexicographically nondecreasing ordering can be found in $O(n^2)$. The dual solution is obtained in n iterations of $O(n)$ calculations. The while statement operates on the square $|I| \times |J|$ submatrix of A and considers each element at most once. Since $|J| \leq n$ we find that the total complexity of this algorithm is $O(n^2)$.

4.2. The minimum cost operating problem

ALGORITHM 4.2.1.

```
begin make of the tree a rooted tree;
    construct a lexicographically nondecreasing ordering $\tau$ of the neigh-
    borhood subtrees;
    define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $i \in \tau(j)$;
    use Algorithm 3.8
end
```

EXAMPLE 4.2.2. We use the same tree and neighborhood subtrees as in Example 4.1.2. Therefore the matrix A is the same. We take $d_i = 2$, $i = 1,\ldots,9$. The matrix A and the successive iteration to obtain the dual solution y are

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\hat{c}_1 = 3$, $\hat{c}_2 = 7$, $\hat{c}_3 = 5$, $\hat{c}_4 = 9$, $\hat{c}_5 = 3$, $\hat{c}_6 = 5$, $\hat{c}_7 = 3$.

$y_1 = 2$, $\hat{c}_1 = 1$, $\hat{c}_3 = 3$.

$y_2 = 1$, $I = \{2\}$, $J = \{1\}$, $\hat{c}_1 = 0$, $\hat{c}_3 = 2$.

$y_3 = 0$.

$y_4 = 2$, $\hat{c}_2 = 5$, $\hat{c}_4 = 7$.

$y_5 = 2$, $\hat{c}_2 = 3$, $\hat{c}_4 = 5$.

$y_6 = 2$, $\hat{c}_2 = 1$, $\hat{c}_4 = 3$, $\hat{c}_7 = 1$.

$y_7 = 1$, $I = \{2,7\}$, $J = \{1,7\}$, $\hat{c}_3 = 1$, $\hat{c}_5 = 2$, $\hat{c}_7 = 0$.

$y_8 = 0$.

$y_9 = 0$.

The total value of the solution y is 10. The while statement of Algorithm 3.8 operates on the submatrix of A with rows belonging to I and columns belonging to J.

I\J	1	7
2	1	0
7	0	1

Iteration 1: $O = \{7\}$.

Iteration 2: $O = \{1,7\}$.

Facilities 1 and 7 can not serve clients at vertices 4 and 5. We have a cost of establishing facilities of 6 and a penalty cost of 4. The total operating cost is 10. □

The complexity of this algorithm is the same as the complexity of the algorithm of the minimum cost covering problem, i.e., $O(n^2)$.

4.3. The uncapacitated plant location problem

ALGORITHM 4.3.1.

```
begin make of the tree a rooted tree;
      for each i calculate the set of distances to the facility locations,
      add zero to this set and order the elements in increasing order, say,
      $0 = r_{i1} < r_{i2} < \dots < r_{it(i)}$; define $r_{i,t(i)+1} = \infty$;
      construct a lexicographically nondecreasing ordering $\tau$ of the sets of
      facility locations in the neighborhood subtrees $T_{ik} = \{y \in T \mid d(v_i,y) \leq r_{ik}\}$, $i = 1,\dots,n$, $k = 1,\dots,t(i)$;
      define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $j \in \tau(i)$;
      for each row i, corresponding say to $T_{jk}$, define $d_i = w_j(r_{j,k+1} - r_{jk})$;
      use Algorithm 3.8
end
```

EXAMPLE 4.3.2. Consider the rooted tree of Figure 4.3.3. The facility locations are 1, 3, and 4 with a cost of 5, 6 and 4 respectively. We assume $w_i = 1$, $i = 1,\dots,5$.

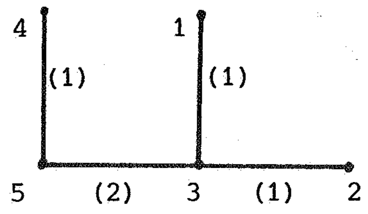

Figure 4.3.3. Rooted tree of example

The distance r_{ik}, the corresponding facility locations in T_{ik} and the value $\hat{d}_{ik} = w_i(r_{i,k+1} - r_{ik})$ are given in Figure 4.3.4.

vertex i	distance r_{ik}	T_{ik}	$\hat{d}_{ik}$
1	0	{1}	1
	1	{1,3}	3
	4	{1,3,4}	∞
2	0	$\emptyset$	1
	1	{3}	1
	2	{1,3}	2
	4	{1,3,4}	∞
3	0	{3}	1
	1	{1,3}	2
	3	{1,3,4}	∞
4	0	{4}	3
	3	{3,4}	1
	4	{1,3,4}	∞
5	0	$\emptyset$	1
	1	{4}	1
	2	{4,3}	1
	3	{4,3,1}	∞

Figure 4.3.4. Data of the example

The matrix A, the vector d and the successive iterations to obtain the dual solution y are

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad d = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 2 \\ 2 \\ 3 \\ 1 \\ 1 \\ 1 \\ \infty \\ \infty \\ \infty \\ \infty \\ \infty \end{bmatrix}$$

$\hat{c}_1 = 5$, $\hat{c}_3 = 6$, $\hat{c}_4 = 4$.
$y_1 = 1$.
$y_2 = 1$.
$y_3 = 1$, $\hat{c}_1 = 4$.
$y_4 = 1$, $\hat{c}_3 = 5$.
$y_5 = 1$, $\hat{c}_3 = 4$.
$y_6 = 3$, $\hat{c}_1 = 1$, $\hat{c}_3 = 1$.
$y_7 = 1$, $I = \{7\}$, $J = \{3\}$, $\hat{c}_1 = 0$, $\hat{c}_3 = 0$.
$y_8 = 0$.
$y_9 = 3$, $\hat{c}_4 = 1$.
$y_{10} = 1$, $I = \{7,10\}$, $J = \{3,4\}$, $\hat{c}_4 = 0$.

all other values are zero

The total value of the dual solution is 13. The while statement of Algorithm 3.8 operates on the submatrix of A with rows belonging to I and columns belonging to J.

I\J	3	4
7	1	0
10	0	1

Iteration 1: $0 = \{4\}$.

Iteration 2: $0 = \{3,4\}$.

The cost of establishing facilities at vertices 3 and 4 is 10. Vertex 5 is supplied by 4 at a cost of 1, vertices 1 and 2 are supplied at a cost of 1 each. The total cost is 13. □

Calculating all distances from a given vertex and order them in increasing order takes $O(n \log n)$ time. Finding all sets corresponding to the rows of A therefore takes $O(n^2 \log n)$ time. Since there are at most n^2 rows we can find a lexicographically nondecreasing ordering in $O(n^3)$ time and also can find the dual solution in $O(n^3)$ time. The while statement operates on the square $|I| \times |J|$ submatrix and therefore takes $O(n^2)$ time. The overall complexity is $O(n^3)$.

5. INTERPRETATION OF DUAL PROBLEMS

In this section we formulate a dual problem of each of the location problems and describe an economic interpretation of the dual problems. This interpretation was first observed by TAMIR [9] in the context of a tree location problem similar to the minimum cost covering problem. In this tree location problem each facility can serve all clients but each client must be served by a facility within a given distance from that client. This problem can be solved in $O(n^2)$ time using the procedure of Section 3 as compared with the $O(n^3)$ dynamic programming algorithm given by TAMIR [8].

The dual problem of the minimum cost covering and operating problem is defined to be the dual of the LP-relaxation of the problem. The dual of the LP-relaxation of the uncapacitated plant location problem (2.5) is given by

$$(5.1) \qquad \max \sum_{i=1}^{n} y_i$$

$$\text{s.t. } y_i - \mu_{ij} \le w_i d(v_i,s_j), \quad i = 1,2,\dots,n, \ j = 1,2,\dots,m,$$

$$\sum_{i=1}^{n} \mu_{ij} \le c_j, \qquad j = 1,2,\dots,m.$$

$$\mu_{ij} \ge 0, \qquad i = 1,2,\dots,n, \ j = 1,2,\dots,m.$$

Given y_i, $i = 1,2,\dots,n$ we can take $\mu_{ij} = \max(0, y_i - w_i d(v_i,s_j))$. Since $y_i \ge 0$ in an optimal solution to (5.1) we get the following restriction of (5.1) which we define to be the dual problem of the uncapacitated plant location problem.

$$(5.2) \qquad \max \sum_{i=1}^{n} y_i$$

$$\text{s.t. } \sum_{i=1}^{n} \max(0, y_i - w_i d(v_i,s_j)) \le c_j, \quad j = 1,2,\dots,m,$$

$$y_i \ge 0, \qquad i = 1,2,\dots,n.$$

For the two covering problems it was shown in Section 3 that we have strong duality. The optimal dual variables of (5.2) are not available when we solve the uncapacitated plant location problem by Algorithm 4.3.1. Since we used formulation (2.9) of the problem Algorithm 4.3.1 obtains an optimal solution y_{ik}, $i = 1,\dots,n$, $k = 1,\dots,t(i)$ of the dual problem of the LP-relaxation of (2.9) given by

$$(5.3)\qquad \begin{aligned} \max \sum_{i=1}^{n} \sum_{k=1}^{t(i)} & y_{ik} \\ \text{s.t.} \sum_{i=1}^{n} \sum_{k=1}^{t(i)} & y_{ik} a_{ij}^{k} \le c_j, \; j = 1,2,\dots,m, \\ & y_{ik} \ge 0, \qquad i = 1,2,\dots,n, \; k = 1,2,\dots,t(i). \end{aligned}$$

We will show in Lemma 5.8 that an optimal solution to y_i, $i = 1,2,\dots,n$ of (5.2) can be obtained from an optimal solution y_{ik}, $i = 1,2,\dots,n$, $k = 1,2,\dots,t(i)$ of (5.3) by defining $y_i = \sum_{k=1}^{t(i)} y_{ik}$, $i = 1,2,\dots,n$. Therefore we also have strong duality for the uncapacitated plant location problem. These dual problems have a nice interpretation which we will now discuss.

Suppose we want to ask client i a contribution y_i in the total cost of serving all clients. In order for a contribution to be acceptable by all clients we can not charge any subset of clients more than it would cost to supply only those clients. If we would charge more for a particular subset of clients, then they may decide to form a coalition since this would reduce their contribution as a group. The problem we consider is to maximize the total contribution subject to the condition that no subject of clients can benefit from forming a coalition. We shall prove that this problem is exactly the dual problem of the corresponding location problem. Since we have strong duality the total contribution is equal to the total cost. Let us denote by $v(I)$ the minimal total cost if only clients $I \subseteq N = \{1,\dots,n\}$ have to be served. For example in the minimum cost covering problem we have

$$\begin{aligned} v(I) = \min \sum_{j=1}^{m} & c_j x_j \\ \text{s.t.} \sum_{j=1}^{m} & a_{ij} x_j \ge 1, \; i \in I \\ & x_j \in \{0,1\}, \; j = 1,2,\dots,m. \end{aligned}$$

The *contribution problem* is given by

$$(5.4) \qquad \max \sum_{i=1}^{n} y_i$$

$$\text{s.t.} \sum_{i\in I} y_i \le v(I), \text{ for all } I \subseteq N,$$

$$y_i \ge 0, \qquad i = 1,2,\dots,n.$$

The rest of this section is devoted to proving that the contribution problem is equal to the dual problem of the corresponding location problem as well as to show how the optimal variables of (5.2) can be obtained from the optimal variables of (5.3).

LEMMA 5.5. *The contribution problem and the dual vroblem corresponding to the minimum cost covering problem are identical.*

PROOF. It is sufficient to prove that the set of feasible solutions are identical. Let y be a dual feasible solution and let $I \subseteq N$. Suppose $v(I) = \sum_{j\in J} c_j$, where $J \subseteq \{1,2,\dots,m\}$. Then there is a (0,1)-solution x such that

$$\sum_{j=1}^{m} a_{ij}x_j = \sum_{j\in J} a_{ij} \ge 1 \text{ for all } i \in I.$$

Then we have

$$\sum_{i\in I} y_i \le \sum_{i\in I} y_i(\sum_{j\in J} a_{ij}) \le \sum_{j\in J}\sum_{i=1}^{n} y_i a_{ij} \le \sum_{j\in J} c_j = v(I).$$

Hence y is feasible in the contribution problem.

Let y be feasible in the contribution problem and let $j \in \{1,2,\dots,m\}$. Define $I = \{i \mid a_{ij} = 1\}$. Then we have

$$\sum_{i\in I} y_i = \sum_{i=1}^{n} y_i a_{ij} \le v(I) \le c_j,$$

where $v(I) \le c_j$ since facility j can serve all clients in I. We conclude that y is a dual feasible solution. □

LEMMA 5.6. *The contribution problem and the dual problem corresponding to the minimum cost operating problem are identical.*

PROOF. Let y be a dual feasible solution and let $I \subseteq N$. Suppose $v(I) = \sum_{j\in J} c_j + \sum_{i\in K} d_i$, where $J \subseteq \{1,2,\ldots,m\}$ and $K \subseteq I$. Then there is a (0,1)-solution x such that

$$\sum_{j=1}^{m} a_{ij}x_j = \sum_{j\in J} a_{ij} \geq 1 \text{ for all } i \in I\backslash K$$

and

$$\sum_{j=1}^{m} a_{ij}x_j = \sum_{j\in J} a_{ij} = 0 \text{ for all } i \in K.$$

Then we have

$$\sum_{i\in I\backslash K} y_i \leq \sum_{i\in I\backslash K} y_i(\sum_{j\in J} a_{ij}) \leq \sum_{j\in J} \sum_{i=1}^{n} y_i a_{ij} \leq \sum_{j\in J} c_j$$

and

$$\sum_{i\in K} y_i \leq \sum_{i\in K} d_i.$$

Hence

$$\sum_{i\in I} y_i \leq \sum_{j\in J} c_j + \sum_{i\in K} d_i = v(I)$$

and y is feasible in the contribution problem.

Let y be feasible in the contribution problem and let $j \in \{1,2,\ldots,m\}$ and $i \in \{1,2,\ldots,n\}$. Define $I_2 = \{i \mid a_{ij} = 1\}$ and $I_2 = \{i\}$. Then we have

$$\sum_{i=1}^{n} y_i a_{ij} = \sum_{i\in I_1} y_i \leq v(I_1) \leq c_j$$

and

$$y_i \leq v(I_2) \leq d_i,$$

where $v(\{i\}) \leq d_i$ since not serving client i is a feasible solution with cost d_i. We conclude that y is a dual feasible solution. □

LEMMA 5.7. *The contribution problem and the dual problem corresponding to the uncapacitated plant location problem are identical.*

PROOF. Let y be a dual feasible solution and let $I \subseteq N$. Suppose $v(I) = \sum_{j\in J} c_j + \sum_{i\in I} w_i d_{ij(i)}$, where $J \subseteq \{1,2,\dots,m\}$ and $j(i)$ is a vertex in J closest to i. Then we have

$$\sum_{i\in I}(y_i - w_i d_{ij(i)}) \le \sum_{i\in I}\max(0,y_i - w_i d_{ij(i)}) \le$$

$$\sum_{j\in J}\sum_{i\in I}\max(0,y_i - w_i d_{ij}) \le \sum_{j\in J}\sum_{i=1}^{n}\max(0,y_i - w_i d_{ij}) \le \sum_{j\in J} c_j$$

or

$$\sum_{i\in I} y_i \le \sum_{j\in J} c_j + \sum_{i\in I} w_i d_{ij(i)} = v(I).$$

Hence y is feasible in the contribution problem.

Let y be feasible in the contribution problem and let $j \in \{1,2,\dots,m\}$. Define $I = \{i \mid y_i \ge w_i d_{ij}\}$. Then we have

$$\sum_{i\in I} y_i \le v(I) \le c_j + \sum_{i\in I} w_i d_{ij}$$

or

$$\sum_{i\in I}(y_i - w_i d_{ij}) \le c_j.$$

By definition of I this is equivalent to

$$\sum_{i=1}^{n}\max(0,y_i - w_i d_{ij}) \le c_j.$$

Hence y is a dual feasible solution. □

LEMMA 5.8. *Let* y_{ik}, $i = 1,\dots,n$, $k = 1,\dots,t(i)$ *be an optimal solution of* (5.3) *obtained by Algorithm* 4.3.1. *Then* $y_i = \sum_{k=1}^{t(i)} y_{ik}$, $i = 1,\dots,n$ *is an optimal solution of* (5.2).

PROOF. Let $y_{ik}, i=1,\dots,n, k=1,\dots,t(i)$ be an optimal solution of (5.3) obtained by Algorithm 4.3.1. In order to prove that $y_i = \sum_{k=1}^{t(i)} y_{ik}$, $i=1,\dots,n$ is an optimal solution of (5.2) it is sufficient to prove that it is a feasible solution since optimality follows from the strong duality result of the solution $y_{ik}, i=1,\dots,n, k=1,\dots,t(i)$. Let $j \in \{1,\dots,m\}$. We shall prove that $\sum_{k=1}^{t(i)} y_{ik} a_{ij}^{k} = \max(0,y_i - w_i d_{ij})$. We then have $\sum_{i=1}^{n}\max(0,y_i - w_i d_{ij}) =$

$\sum_{i=1}^{n} \sum_{k=1}^{t(i)} y_{ik} a_{ij}^{k} \leq c_j$. Let $\ell_1 \in \{1,\ldots,t(i)\}$ be such that $r_{i\ell_1} = d_{ij}$. Then $a_{ij}^{k} = 1$, $k \geq \ell_1$ and $a_{ij}^{k} = 0$ otherwise. Therefore $\sum_{k=1}^{t(i)} y_{ik} a_{ij}^{k} = \sum_{k=\ell_1}^{t(i)} y_{ik}$. Let $\ell_2 \in \{1,\ldots,t(i)\}$ be the first index such that $0 \leq y_{i\ell_2} < w_i(r_{i,\ell_2+1} - r_{i\ell_2})$. Since in Algorithm 4.3.1 y_{ik} is calculated before $y_{i\ell}$ is calculated $(k < \ell)$ and taken to be as large as possible with respect to the dual constraints we have $y_{ik} = w_i(r_{i,k+1} - r_{ik})$ for all k, $k < \ell_2$ and $y_{ik} = 0$ for all k, $k \geq \ell_2 + 1$. Then we have

$$y_i = \sum_{k=1}^{t(i)} y_{ik} = \sum_{k=1}^{\ell_2} y_{ik}.$$

If $\ell_2+1 \leq \ell_1$, then $y_i < w_i r_{i\ell_1} = w_i d_{ij}$ and $\sum_{k=\ell_1}^{t(i)} y_{ik} = 0$.
If $\ell_2+1 > \ell_1$, then $\sum_{k=\ell_1}^{t(i)} y_{ik} = \sum_{k=\ell_1}^{\ell_2} y_{ik} = y_i - \sum_{k=1}^{\ell_1-1} y_{ik} = y_i - w_i r_{i\ell_1} = y_i - w_i d_{ij}$.
We conclude that $\sum_{k=1}^{t(i)} y_{ik} a_{ij}^{k} = \max(0, y_i - w_i d_{ij})$. □

APPENDIX

In this appendix we prove our main result stated in Theorem 3.5. We will need the following well-known properties of trees. Proofs of the properties can be found for example in KOLEN [6].

Property 1. If x, y and z are points on a tree, then there is a point $t \in P(x,y) \cap P(y,z) \cap P(z,x)$.

Property 2. If x, y and z are points on a tree such that $z \in P(x,y)$, then for all points t on the tree we have $z \in P(x,t)$ or $z \in P(y,t)$.

Property 3. If v is a given vertex of a tree and t is defined to be a vertex at largest distance from v, then t is endpoint of a longest path in the tree.

Property 4. Let $P(t_1,t_2)$ be a longest path in the tree. If $x \in T$ and $r \geq 0$ such that $d(x,t_i) \leq r$, $i = 1,2$, then $\{y \in T \mid d(y,x) \leq r\} = T$.

We can now prove Theorem 3.5 which is restated in the next theorem.

<u>THEOREM 5</u>. *Let* $p(t_1,t_2)$ *be a longest path in the tree. Let* $E_1 = \{y \in T \mid d(y,x_1) \le r_1\}$ *and* $E_2 = \{y \in T \mid d(y,x_2) \le r_2\}$ *be neighborhood subtrees containing* t_1. *Then* $E_1 \subseteq E_2$ *or* $E_2 \subseteq E_1$.

<u>PROOF</u>. Define $s_i \in E_i$ to be the point on $P(t_1,t_2)$ closest to t_2. Without loss of generality assume that $d(t_2,s_2) \le d(t_2,s_1)$. We shall prove that $E_1 \subseteq E_2$. If $s_2 = t_2$, then it follows from Property 4 that $E_2 = T$ and hence $E_1 \subseteq E_2$. Therefore we may assume that $s_2 \ne t_2$. Let $p \in E_1$ We shall prove that $p \in E_2$. According to Property 1 there is a point $q \in P(p,t_1) \cap P(p,s_1) \cap P(s_1,t_1)$ (Note that $q \in P(p,s_1) \subseteq E_1$). We shall prove in Lemma 6 that $d(p,q) \le d(q,t_1)$ and $d(p,q) \le d(q,s_1)$. Since $q \in P(t_1,s_1)$ we have according to Property 2. $q \in P(t_1,x_2)$ or $q \in P(s_1,x_2)$.

<u>Case 1</u>: $q \in P(x_2,t_1)$. Then

$$\begin{aligned} d(p,x_2) &\le d(p,q) + d(q,x_2) \\ &\le d(t_1,q) + d(q,x_2) \quad \text{(Lemma 6)} \\ &= d(t_1,x_2) \qquad (q \in P(x_2,t_1)) \\ &\le r_2 \qquad\qquad (t_1 \in E_2). \end{aligned}$$

<u>Case 2</u>: $q \in P(s_1,x_2)$. Then

$$\begin{aligned} d(p,x_2) &\le d(p,q) + d(q,x_2) \\ &\le d(s_1,q) + d(q,x_2) \quad \text{(Lemma 6)} \\ &= d(s_1,x_2) \qquad (q \in P(s_1,x_2)) \\ &\le r_2 \qquad\qquad (s_1 \in P(t_1 s_1) \subseteq P(t_1,s_2) \subseteq E_2). \end{aligned}$$

It follows that $p \in E_2$. Hence $E_1 \subseteq E_2$. □

<u>LEMMA 6</u>. *Let* s_1,s_2,t_1,t_2,p,q *be defined as in Theorem 5. Then* $d(p,q) \le d(q,t_1)$ *and* $d(p,q) \le d(q,s_1)$.

<u>PROOF</u>. First of all we prove that

(7) $\qquad s_1 \in P(y,t_2)$ for all $y \in E_1$.

Let $y \in E_1$. Define z to be a point at $P(y,s_1) \cap P(s_1,t_2) \cap P(y,t_2)$. Since $z \in P(y,s_1)$ we have $z \in E_1$. Since $z \in P(s_1,t_2)$ and $z \in E_1$ we have by definition of s_1 that $z = s_1$. We conclude that $s_1 \in P(y,t_2)$. It follows from (7) that $s_1 \in P(x_1,t_2)$ and by definition of s_1 we have $d(x_1,s_1) = r_1$. We have

$$d(t_1,t_2) = d(t_1,q) + d(q,t_2) \qquad (q \in P(t_1,s_1) \subseteq P(t_1,t_2))$$

and

$$\begin{aligned} d(p,t_2) &= d(p,s_1) + d(s_1,t_2) && (s_1 \in P(p,t_2) \quad (7)) \\ &= d(p,q) + d(q,s_1) + d(s_1,t_2) && (q \in P(p,s_1)) \\ &= d(p.q) + d(q,t_2) && (s_1 \in P(q,t_2) \quad (7)). \end{aligned}$$

Since $P(t_1,t_2)$ is a longest path in the tree we have $d(t_1,t_2) \geq d(p,t_2)$ and hence $d(q,t_1) \geq d(p,q)$.

Since $q \in P(p,t_1)$ we have $q \in P(t_1,x_1)$ or $q \in P(p,x_1)$. In order to prove that $d(p,q) \leq d(q,s_1)$ we consider these two cases.

Case 1: $q \in P(x_1,t_1)$. Then

$$\begin{aligned} d(q,t_1) &= d(x_1,t_1) - d(q,x_1) && (q \in P(x_1,t_1)) \\ &\leq r_1 - d(q,x_1) && (t_1 \in E_1) \\ &= d(x_1,s_1) - d(q,x_1) && (d(x_1,s_1) = r_1) \\ &\leq d(q,s_1). \end{aligned}$$

Since $d(p,q) \leq d(q,t_1)$ we have $d(p,q) \leq d(q,s_1)$.

Case 2: $q \in P(p,x_1)$. Then

$$\begin{aligned} d(p,q) &= d(p,x_1) - d(q,x_1) && (q \in P(p,x_1)) \\ &\leq r_1 - d(q,x_1) && (p \in E_1) \\ &= d(x_1,s_1) - d(q,x_1) && (d(x_1,s_1) = r_1) \\ &\leq d(q,s_1). \quad \square \end{aligned}$$

REFERENCES

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

[2] BILDE, O. & J. KRARUP (1967) Bestemmelse af optimal beliggenhed af produktionssteder, Research Report, IMSOR, Danmarks Tekniske Højskole.

[3] ERLENKOTTER, O. (1978) A dual-based procedure for uncapacitated facility location. *Oper. Res.26*,992-1009.

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

[5] HOFFMAN, A.J., A. KOLEN & M. SAKAROVITCH (1982) Totally-balanced and greedy matrices. To appear in *SIAM J. Algebraic and Discrete Methods*.

[6] KOLEN, A. (1982) *Location Problems on Trees and in the Rectilinear Plane*, Ph. D. thesis, Mathematisch Centrum, Amsterdam.

[7] KRARUP J. & P.M. PRUZAN (1977) Selected families of discrete location problems. Part III: The plant location family. Working Paper no WP-12-77. University of Calgary.

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

[9] TAMIR, A (1980) On the core of cost allocation games defined on location problems. Report Department of Statistics, Tel-Aviv University.

[10] TAMIR, A (1981) Personal communication (April 1981, Tel-Aviv University).