6

Covering Problems

Antoon Kolen
Department of Quantitative Economics
University of Limburg
Maastricht, The Netherlands

Arie Tamir
Department of Statistics
Tel Aviv University
Ramat Aviv, Tel Aviv, Israel

6.1. INTRODUCTION

A class of location decision problems may be formulated as *covering problems*. We will study in this chapter some covering problems on *networks*. Here a set of clients, with given demands for services or commodities, are assumed to be located on the nodes or arcs of an underlying network. The problems require that facilities be established on the network to provide the necessary services or commodities for these clients. All facilities will be assumed to be of the same type and each of them to have sufficient capacity to serve all clients; that is, only the *uncapacitated* versions of the covering problems will be considered. Whenever a facility is "close enough," as defined by the problem statement, to serve a client we say that the client is *covered* by that facility. Generally, in the covering problems discussed here we determine the number and locations of the facilities so

that the clients are covered with minimum cost, the cost function (or objective function) being defined by the problem statement.

In this introduction we give an informal description of the various location problems to be discussed in this chapter. Most of these problems are either covering problems or can be solved by a sequence of covering problems. Exact formulations are given in the succeeding sections where we present solution techniques.

In most of our models the clients are assumed to be located only at the nodes of the network. We refer to the case where clients are not restricted to be at the nodes as a *continuous location problem*. A point in the network where a facility may be established is referred to as a *potential site*.

We consider three types of costs involved with a given set of facility locations. The first is the *setup cost*, that is, the cost of establishing a facility at a given site. When the facilities may be located anywhere along the network we assume that all setup costs are equal. When the set of potential sites is finite, different setups will also be allowed. The second cost is the *transportation cost*, that is, the cost of transporting the service (or commodity) between a facility and a client. The transportation cost is assumed to be a nondecreasing function of the distance between the client and the serving facility. Since we have assumed that each facility has a sufficient capacity to serve all clients, each client will be served by its closest facility. Finally, we may have what we refer to as the *penalty cost*, a cost which is applied only if a client is not served by any facility. The penalty costs will, in general, depend on the particular clients not served.

The following constraints are also included in our models. The first is a *budget constraint* which models the upper bound on the total setup cost. If all setup costs are equal, then the budget constraint reduces to an upper bound on the number of facilities that may be established.

In addition we have *client constraints* and *facility constraints*. Through the client constraints, we model the requirement that each client be served by a facility which is located within a given distance (depending on the client) from that client. One can think of the set of points within a given distance from a client as the *region* of *attraction* for that client. Consider, for example, the problem of a new company which is interested in entering an existing market. We may assume that a client will switch to the new company only if the latter establishes a facility within his region of attraction.

The facility constraint refers to the case when a facility can only serve clients which are located within a given distance (depending on the facility) from it. Here we may assume that associated with clients who are located at large distances, are transportation costs that are too expensive for the serving company.

Common objective functions that arise are:

1. minimize the maximum transportation cost (the *center problem*);
2. minimize the sum of the transportation costs (the *median problem*);

3. minimize the sum of the transportation costs and the setup costs (the *uncapacitated facility location problem*);
4. minimize the sum of the setup costs and penalty costs (the *covering problem*);
5. minimize the penalty costs (the *coverage problem*).

In Section 6.2 we formulate both the client and facility constrained problems as (0, 1) integer programming problems. We present a polynomial time algorithm to solve this problem whenever the integer programming problem satisfies certain properties. These properties are shown to be satisfied when the underlying network has a tree structure. We also show that the uncapacitated facility location (UFL) problem can be formulated as a client constrained covering problem. Hence, it can be solved in polynomial time in the case of a tree network by using the algorithm presented.

One aspect of location problems which is usually not addressed is the aspect of allocating the total locational cost among the clients. In Section 6.2 we discuss the problem of finding a cost allocation which is "acceptable" to all clients. After defining what is meant by "acceptable," we develop a relationship between the cost allocation problem and the dual of the covering problem. Acceptable allocations are shown to always exist for tree networks.

In the context of covering problems we also discuss the *multiple covering problem*. Here, each client must be served by a number (depending on the client) of facilities. There is also an upper bound (depending on the site) on the number of facilities we can establish at a given site. We show how the algorithm developed in Section 6.2 can be used to solve the multiple covering problem on tree networks when all setup costs are equal.

Section 6.3 is devoted to budget constrained center problems on trees. We show that for different setup costs this problem can be solved as a sequence of covering problems. In the case of equal setup costs, the budget constraint specifies the number of facilities, say p; in this case we obtain the classical p-center problem. Using the concept of a chordal graph we establish a duality result for a class of p-center problems.

An extension of the classical center problem is the *round-trip center problem*. In this problem we consider pairs of clients and a transportation company which has to transport services (or commodities) from one client to the other. A facility in this case is a depot where the vehicles of the company are to be located. The transportation cost is assumed to be a monotone nondecreasing function of the *round-trip distance*, that is, the distance a vehicle has to travel to deliver the service from one client to another, and then to return to its depot. Equivalently, the round trip distance is the sum of the distances between the depot and the two clients and the distance between the two clients. We prove that, unlike the center problem, the round-trip version is $\mathcal{NP}$*-hard* for different setup costs, even on tree networks.

When the facilities can be located anywhere along the tree, and the setup costs are equal, the round-trip p-center problem, and the p-center problem, can be solved in polynomial time. When we restrict the potential sites to the set of all nodes, again we can solve both problems in polynomial time. However, in the case when we restrict the sites to a strict subset of the nodes, the p-center problem can still be solved in polynomial time, while the round-trip p-center problem becomes $\mathcal{NP}$-hard.

In Section 6.4 we show that there is a strong relationship between the location problems and set covering problems on totally balanced matrices. We show that the solution technique for the covering problem developed in Section 6.2 is equally applicable to the case where the $(0, 1)$ matrix of the covering problem is totally balanced. We also discuss the budget constrained coverage problem with equal setup costs and the p-median problem, both on tree networks. These problems are special cases of integer programming problems on totally balanced matrices, and are also solvable in polynomial time.

6.2. THE INTEGER PROGRAMMING APPROACH

As a general framework for the location problems we utilize the following integer covering program:

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in M_2} c_j x_j + \sum_{\omega \in M_1} p_\omega z_\omega \\ \text{subject to} \quad & \sum_{j \in M_2} a_{\omega j} x_j + z_\omega \geq 1, \qquad \forall \omega \in M_1, \qquad (6.2.1) \\ & x_j \in \{0, 1\}, \qquad j \in M_2, \\ & z_\omega \in \{0, 1\}, \qquad \omega \in M_1, \end{aligned}$$

where M_1, M_2 are finite index sets and $A = (a_{\omega j})$ is an $|M_1| \times |M_2|$ $(0, 1)$ matrix. Let $N = (V, E)$ be an undirected network with node set $V = \{v_1, v_2, \ldots, v_m\}$ and arc set E. Each arc $a \in E$ has a certain length $\alpha_a \geq 0$. If we consider each arc $a = [v_i, v_j]$ as a line segment of length α_a, then we can define a point on the arc by its distance on the segment from v_i. We define the *distance* $d(x, y)$ between two points x and y on N to be the length of a shortest path (denoted by $P[x, y]$) from x to y. The *distance* $D(y, X)$ between a point y on N and a closed set of points X on N is defined to be the distance $d(y, x)$ where x is a closest point to y in the set X. We also let $D(X, y)$ denote the distance $D(y, X)$. We assume that at each node there exists exactly one client. We refer to the client located at node v_i as *client* i, $i = 1, 2, \ldots, m$. If client i is not served by any facility, then a nonnegative penalty cost of p_i is incurred. We assume that the set of potential sites for the facilities is a subset of nodes of the network. Without loss of generality

let this set be $V' = \{v_1, \ldots, v_n\}$, $n \leq m$. The nonnegative setup cost of establishing a facility at v_j is c_j, $j = 1, \ldots, n$. (We will refer to a facility established at site v_j as *facility* j, $j = 1, \ldots, n$.) The *covering problem* is to minimize the sum of setup costs and penalty costs, and it corresponds to the special case of the covering program (6.2.1) with $M_1 = \{1, \ldots, m\}$, $M_2 = \{1, \ldots, n\}$ and $a_{ij} = 1$ if and only if a facility at site v_j can serve client i. Thus, the covering problem is formulated as,

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} p_i z_i \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j + z_i \geq 1, \qquad i = 1, \ldots, m, \qquad (6.2.2) \\ & x_j \in \{0, 1\}, \qquad j = 1, \ldots, n, \\ & z_i \in \{0, 1\}, \qquad i = 1, \ldots, m, \end{aligned}$$

where $x_j = 1$ if and only if a facility is established at v_j and $z_i = 1$ if and only if client i is served by no facility.

The *client constrained covering problem* corresponds to the case where we have a region of attraction of radius r_i for client i and we set $a_{ij} = 1$ if and only if $d(v_i, v_j) \leq r_i$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

The *facility constrained covering problem* corresponds to the case where we have a radius s_j for facility j and we set $a_{ij} = 1$ if and only if $d(v_i, v_j) \leq s_j$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

The *uncapacitated facility location problem* (UFL) is to locate facilities at the nodes in order to minimize the sum of the setup and transportation costs. The transportation cost for client i is given by a nondecreasing function $f_i(d)$ of the distance d between client i and the facility serving it. We let $f_i(\infty)$ correspond to the case when client i is served by no facility. Due to the fact that facilities are uncapacitated and the transportation costs are nondecreasing, we will assume without loss of generality that each client is served by its closest facility. If S is the subset of nodes corresponding to established facilities, then the corresponding objective value of the UFL problem is given by

$$\sum_{j:\, v_j \in S} c_j + \sum_{i=1}^{m} f_i(D(v_i, S)) \qquad (6.2.3)$$

where by convention $D(v_i, \varphi) = \infty$.

Let us return for a moment to the client constrained covering problem (that is, the case where $a_{ij} = 1$ if and only if $d(v_i, v_j) \leq r_i$), and minimize (6.2.2) for a given solution $x_j \in \{0, 1\}$, $j = 1, \ldots, n$. Define $S = \{v_j : x_j = 1\}$. Since $p_i \geq 0$ an optimal solution is given by $z_i = 1$ if

$$\sum_{j=1}^{n} a_{ij} x_j = 0 \quad \text{(or equivalently } D(v_i, S) > r_i\text{)},$$

and $z_i = 0$ if

$$\sum_{j=1}^{n} a_{ij}x_j \geq 1 \quad \text{(or equivalently } D(v_i, S) \leq r_i\text{)}, \qquad i = 1, \ldots, m.$$

If we define the function $f_i(r)$, $i = 1, 2, \ldots, m$ by

$$f_i(r) = \begin{cases} 0, & \text{if } r \leq r_i, \\ p_i, & \text{if } r > r_i, \end{cases} \tag{6.2.4}$$

then the optimal objective value of (6.2.2) corresponding to the set of established facilities S is given by (6.2.3).

We conclude that both the client constrained covering problem and the UFL problem can be formulated as

$$\underset{S \subseteq V'}{\text{minimize}} \left\{ \sum_{j:\, v_j \in S} c_j + \sum_{i=1}^{m} f_i(D(v_i, S)) \right\} \tag{6.2.5}$$

where f_i, $i = 1, \ldots, m$ is a nondecreasing function.

We demonstrate next that the UFL problem can be formulated as a client constrained covering problem. Rewrite (6.2.5) as

$$\begin{aligned} &\text{minimize} && \left[\sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} f_i\left(\min_{\{v_j :\, x_j = 1\}} \{d(v_i, v_j)\} \right) \right] \\ &\text{subject to} && x_j \in \{0, 1\}, \qquad j = 1, \ldots, n \end{aligned} \tag{6.2.6}$$

where $x_j = 1$ if and only if a facility is established at v_j.

By introducing $m \cdot n$ constraints and $m \cdot n$ new (0, 1)-variables we replace the nonlinear part in the objective function of (6.2.6) by a linear expression. The idea behind this transformation is the following. For each client (node) v_i, $i = 1, \ldots, m$, we denote by $r_{i1} \leq r_{i2} \leq \cdots \leq r_{in}$ the (sorted) sequence of distances from v_i to the n potential sites in V'. We also define $r_{i,n+1} = \infty$. If there is no established facility within distance r_{ik} from v_i, then the closest established facility is at a distance of at least $r_{i,k+1}$. In this case the transportation cost for client i is increased from $f_i(r_{ik})$ (the cost if the closest facility was at distance r_{ik}) to at least $f_i(r_{i,k+1})$. Therefore we can consider the difference $f_i(r_{i,k+1}) - f_i(r_{ik})$ as a nonnegative penalty for not establishing a facility within distance r_{ik} from v_i. Define the $(m \cdot n) \times n$ (0, 1) matrix $A = (a_{ik,j})$ by

$$\begin{aligned} a_{ik,j} = 1 \text{ iff } d(v_i, v_j) \leq r_{ik}, \qquad & i = 1, 2, \ldots, m \\ & j, k = 1, 2, \ldots, n. \end{aligned} \tag{6.2.7}$$

We now introduce the (0, 1) variables z_{ik}, where $z_{ik} = 1$ if and only if there is no facility established within distance r_{ik} from v_i, $i = 1, \ldots, m$, $k =$

$1, \ldots, n$. Let x_j, $j = 1, \ldots, n$ be some solution for (6.2.6). Then, we claim that the transportation cost for client i is given by

$$f_i\Big(\min_{\{v_j : x_j = 1\}} \{d(v_i, v_j)\}\Big) = \text{minimize} \sum_{k=1}^{n} (f_i(r_{i,k+1}) - f_i(r_{ik}))z_{ik} + f_i(r_{i1})$$

subject to

$$\sum_{j=1}^{n} a_{ik,j}x_j + z_{ik} \geq 1\,, \qquad k = 1, \ldots, n$$

$$z_{ik} \in \{0, 1\}\,, \qquad k = 1, \ldots, n\,. \tag{6.2.8}$$

To show (6.2.8) observe that since the coefficient of z_{ik} in the objective function of (6.2.8) is nonnegative an optimal solution is obtained by making z_{ik} as small as possible, that is, by making

$$z_{ik} = \begin{cases} 0\,, & \text{if } \sum_{j=1}^{n} a_{ik,j}x_j \geq 1\,, \\ 1\,, & \text{if } \sum_{j=1}^{n} a_{ik,j}x_j = 0\,. \end{cases} \tag{6.2.9}$$

If we let $t(i)$ be the smallest index such that $r_{it(i)} = \min_{\{v_j : x_j = 1\}} \{d(v_i, v_j)\}$ then the number of facilities established within distance r_{ik} from v_i is zero for $k < t(i)$, and at least one for $k \geq t(i)$. Since $\Sigma_{j=1}^{n} a_{ik,j}x_j$ counts the number of established facilities within distance r_{ik} from v_i we have

$$z_{ik} = \begin{cases} 0\,, & \text{if } k \geq t(i)\,, \\ 1\,, & \text{if } k < t(i)\,. \end{cases} \tag{6.2.10}$$

Substituting this in the right-hand side of the objective in (6.2.8) validates our claim:

$$\sum_{k=1}^{t(i)-1} (f_i(r_{i,k+1}) - f_i(r_{ik})) + f_i(r_{i1}) = f_i(r_{it(i)}) = f_i\Big(\min_{\{v_j : x_j = 1\}} \{d(v_i, v_j)\}\Big)\,. \tag{6.2.11}$$

From (6.2.6) and (6.2.8) we obtain the following (0, 1) integer programming formulation of the UFL problem

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \sum_{k=1}^{n} (f_i(r_{i,k+1}) - f_i(r_{ik}))z_{ik} + \sum_{i=1}^{m} f_i(r_{i1})$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ik,j}x_j + z_{ik} \geq 1, \qquad i = 1, \ldots, m, \quad k = 1, \ldots, n$$

$$z_{ik} \in \{0, 1\}, \qquad i = 1, \ldots, m, \quad k = 1, \ldots, n$$

$$x_j \in \{0, 1\}, \qquad j = 1, \ldots, n. \tag{6.2.12}$$

Formulation (6.2.12) of the UFL problem is a client constrained covering problem (note that $a_{ik,j} = 1$ if and only if $d(v_i, v_j) \leq r_{ik}$).

Let us now consider the special case when the network is a tree. Let us denote a tree by $T = (V, E)$. A *subtree* is a closed connected subset of points of T (closedness is meant with respect to the metric induced by the distance function). A *neighborhood subtree* $N(x, r)$ is defined as the set of all points on the tree within a distance r (called the *radius*) from x (called the *center*), that is, $N(x, r) = \{y \in T: d(y, x) \leq r\}$. A neighborhood subtree of a tree is a generalization of an interval on a line segment. We will see that neighborhood subtrees possess some of the useful properties of intervals.

An *intersection matrix* of the set $\{S_1, \ldots, S_m\}$ versus $\{R_1, \ldots, R_n\}$, where S_i, $i = 1, \ldots, m$, and R_j, $j = 1, \ldots, n$ are subsets of a given set is defined to be the $m \times n$ (0, 1) matrix $A = (a_{ij})$ defined by $a_{ij} = 1$ if and only if $S_i \cap R_j$ is nonempty.

If in the client constrained covering problem we define a neighborhood subtree $S_i = \{y \in T: d(v_i, y) \leq r_i\}$, $i = 1, \ldots, m$ then the matrix $A = (a_{ij})$ of (6.2.2) is the intersection matrix of neighborhood subtrees versus nodes, that is, $a_{ij} = 1$ if and only if $v_j \in S_i$.

Similarly we can define $R_j = \{y \in T: d(y, v_j) \leq s_j\}$, $j = 1, \ldots, n$. In this case the matrix $A = (a_{ij})$ of (6.2.2) is the intersection matrix of nodes versus neighborhood subtrees, that is, $a_{ij} = 1$ if and only if $v_i \in R_j$.

We shall prove in Section 6.2.2 that for intersection matrices of nodes versus neighborhood subtrees of a tree there exists a permutation of the rows and a permutation of the columns (we give an efficient procedure to find these permutations), such that the transformed matrix does not contain a 2×2 submatrix of the form

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}. \tag{6.2.13}$$

A matrix that does not contain the above submatrix is said to be in *standard greedy form*. Since a matrix is in standard greedy form if and only if its transpose is in standard greedy form we can also transform the intersection matrix of neighborhood subtrees versus nodes into standard greedy form.

In Section 6.2.1 we give an $O(mn)$ algorithm to solve the covering problem on an $m \times n$ matrix in standard greedy form. We also define and solve the multiple covering problem for a matrix in this form.

In Section 6.2.2 we discuss how we can find the permutations which transform the intersection matrix of nodes versus neighborhood subtrees into standard greedy form.

In Section 6.2.3 we define a dual problem to the covering problem and relate it to the allocation of the total cost among the clients.

6.2.1. The Covering Algorithm for Standard Greedy Matrices

Consider the covering problem

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} p_i z_i \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j + z_i \geq 1 , \qquad i = 1, \ldots, m , \\ & x_j \in \{0, 1\} , \qquad j = 1, \ldots, n , \\ & z_i \in \{0, 1\} , \qquad i = 1, \ldots, m \end{aligned} \tag{6.2.14}$$

where $A = (a_{ij})$ is in standard greedy form. Without loss of generality we assume that $c_j > 0$, $j = 1, \ldots, n$ and $p_i > 0$, $i = 1, \ldots, m$; if $c_j = 0$, then we can take $x_j = 1$ and eliminate from (6.2.14) all rows i and variables z_i for which $a_{ij} = 1$; if $p_i = 0$, then we take $z_i = 1$ and eliminate row i from (6.2.14). Let us define the *LP-relaxation* of (6.2.14) as the problem obtained from (6.2.14) by replacing the integrality constraints by the nonnegativity constraints on the variables.

The algorithm we describe shortly starts by solving the dual of the LP-relaxation:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} y_i \\ & \sum_{i=1}^{m} y_i a_{ij} \leq c_j , \qquad j = 1, \ldots, n , \\ & 0 \leq y_i \leq p_i , \qquad i = 1, \ldots, m . \end{aligned} \tag{6.2.15}$$

After a feasible solution y of (6.2.15) is obtained by our algorithm, it constructs a feasible $(0, 1)$-solution x, z of (6.2.14) which, together with y, satisfy the complementary slackness relations given by

$$y_i \left(\sum_{j=1}^{n} a_{ij} x_j + z_i - 1 \right) = 0 , \qquad i = 1, \ldots, m , \tag{6.2.16}$$

$$x_j \left(\sum_{i=1}^{m} y_i a_{ij} - c_j \right) = 0 , \qquad j = 1, \ldots, n , \tag{6.2.17}$$

$$z_i (y_i - p_i) = 0 , \qquad i = 1, \ldots, m . \tag{6.2.18}$$

It follows that x, z is an optimal solution of (6.2.14).

In the first phase, the algorithm starts by calculating $y_1, \ldots, y_m$ recursively by a greedy approach, that is, given $y_1, \ldots, y_{k-1}$ such that $\sum_{i=1}^{k-1} y_i a_{ij} \le c_j$, $j = 1, \ldots, n$ and $0 \le y_i \le p_i$, $i = 1, \ldots, k-1$ we give y_k the largest value such that $\sum_{i=1}^{k} y_i a_{ij} \le c_j$, $j = 1, \ldots, n$ and $0 \le y_k \le p_k$. So $y_1, \ldots, y_m$ is defined recursively by

$$y_k = \min \left\{ p_k, \min_{j : a_{kj}=1} \left\{ c_j - \sum_{i=1}^{k-1} y_i a_{ij} \right\} \right\}, \qquad k = 1, \ldots, m . \tag{6.2.19}$$

If y_k is such that $\sum_{i=1}^{k-1} y_i a_{ij} < c_j$ and $\sum_{i=1}^{k} y_i a_{ij} = c_j$, then we say that constraint j is *saturated* by y_k. Constraint j is a *binding* constraint if $\sum_{i=1}^{m} y_i a_{ij} = c_j$. Index sets I and J play an important role in our algorithm, where I is defined to be the index set of the y-variables which saturate a constraint, and J is defined to be the index set of the binding constraints, that is, $I = \{i \mid y_i \text{ saturates a constraint}\}$ and $J = \{j \mid \text{constraint } j \text{ is binding}\}$.

In the second phase, the algorithm constructs a subset J^* of J. The $(0, 1)$ solution x, z of (6.2.14) is defined by

$$x_j = \begin{cases} 1, & \text{if } j \in J^*, \\ 0, & \text{otherwise}, \end{cases}$$

$$z_i = \begin{cases} 1, & \text{if } \sum_{j \in J^*} a_{ij} = 0, \\ 0, & \text{otherwise}. \end{cases}$$

The subset J^* of J is defined as follows (beginning with $J^* = \varphi$); add the largest index $k \in J$ to J^* and delete all indices $j \in J$ from J for which $a_{ij} = a_{ik} = 1$ for some $i \in I$; repeat until $J = \varphi$.

Clearly x, z, and y are feasible solutions. In order to prove that they are optimal solutions we show that the complementary slackness relations hold. Let us say that row i is *covered* by column j if $a_{ij} = 1$. The following properties, to be proven shortly, are required to show that x, y, and z are optimal.

Property 6.1. Each row $i \in I$ is covered by exactly one column $j \in J^*$.

Property 6.2. Each row $i \notin I$ with $y_i = 0$ is covered by at least one column $j \in J^*$.

Property 6.3. Each row $i \notin I$ with $y_i = p_i$ is covered by at most one column $j \in J^*$.

Theorem 6.1. *Let x, z and y be the solutions defined above. Then x, z and y are optimal solutions of* (6.2.14) *and* (6.2.15), *respectively.*

Proof. Since $J^* \subseteq J$ (6.2.17) is satisfied. From Property 6.1 and the definition of z it follows that $z_i = 0$ for $i \in I$. Equivalently it follows from

Property 6.2 that $z_i = 0$ for $i \notin I$ and $y_i = 0$. From Property 6.3 and the definition of z it follows that $\Sigma_{j \in J^*} a_{ij} + z_i = 1$ for $i \notin I$ and $y_i = p_i$. Hence the complementary slackness relations (6.2.16), (6.2.17), and (6.2.18) hold. □

Proof of Property 6.1. Let j be the largest constraint which is saturated by y_i. If $j \notin J^*$, then j was deleted from J because there exists an index $k \in J^*$, $k > j$ and an index $p \in I$ such that $a_{pj} = a_{pk} = 1$. If $p = i$, then row i is covered by column k. If $p \neq i$, then since y_i saturates constraint j we must have $p < i$. (Note that $y_1, \ldots, y_m$ were defined recursively.) By the standard greedy form $a_{pj} = a_{pk} = a_{ij} = 1$ imply $a_{ik} = 1$. Therefore row i is covered by column $k \in J^*$. Thus each row $i \in I$ is covered at least once. Whenever we add $k \in J$ to J^* all indices $j \in J$ for which $a_{ij} = a_{ik} = 1$ for some $i \in I$, are deleted from J; hence each row $i \in I$ can be covered at most once. We conclude that row $i \in I$ is covered exactly once by a column from J^*. □

Proof of Property 6.2. Since $y_i = 0$ there exists a constraint j with $a_{ij} = 1$ which was saturated by y_p before y_i was calculated by the algorithm, that is, $p < i$. It follows from the proof of Property 6.1 that row p is covered by a column $k \in J^*$ which is at least as large as any $j \in J$ that is saturated by y_p. If $j = k$, then row i is covered by column k, else by the standard greedy form $a_{pj} = a_{pk} = a_{ij} = 1$ imply $a_{ik} = 1$, and again row i is covered by column k. We conclude that row $i \notin I$ with $y_i = 0$ is covered at least once by a column from J^*. □

Proof of Property 6.3. Suppose that row i is covered by the two columns j, $k \in J^*$ with $j < k$. Let column j be saturated by y_p. Since $y_i > 0$ and $i \notin I$ we have $p > i$. By the standard greedy form $a_{ij} = a_{ik} = a_{pj} = 1$ imply $a_{pk} = 1$. Hence row $p \in I$ is covered by columns j, $k \in J^*$, contradicting Property 6.1. We conclude that row $i \notin I$, $y_i = p_i$ is covered at most once by a column from J^*. □

Let us give an example to demonstrate the algorithm.

Example 6.1. The matrix $A = (a_{ij})$ of (6.2.14) is given by

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

We have $c_1 = 2$, $c_2 = 4$, $c_3 = 4$, $c_4 = 4$, $c_5 = 3$, $c_6 = 8$, $p_i = 2$, $i = 1, 2, \ldots, 7$. Using the algorithm we find $y_1 = 2$, $y_2 = 2$, $y_3 = 2$, $y_4 = 0$, $y_5 = 0$, $y_6 = 2$, $y_7 = 1$, $I = \{3, 7\}$ and $J = \{2, 3, 5\}$. Furthermore $x_3 = x_5 = 1$, all other x_j are zero, $z_6 = 1$, all other z_i are zero. The total cost is 9. This covering problem corresponds to the location problem of Example 6.2. ○

Notice that whenever y_i saturates more than one constraint we only need to keep the last one since all others will never be chosen by the algorithm. In the example, y_3 saturated constraints 2 and 3 so we only had to add 3 to J.

Let us now define the *multiple covering problem*. This can be formulated as

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{n} c_j x_j \\ &\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \geq b_i\,, \qquad i = 1, \ldots, m \\ &&& 0 \leq x_j \leq u_j\,, \qquad x_j \text{ integer}\,, \qquad j = 1, \ldots, n\,, \end{aligned} \tag{6.2.20}$$

where b_i, $i = 1, \ldots, m$, u_j, $j = 1, \ldots, n$ are integers. Each client i has to be served by b_i facilities, $i = 1, \ldots, m$, and there may exist at most u_j facilities at location j, $j = 1, \ldots, n$. A feasible solution exists if and only if $\sum_{j=1}^{n} a_{ij} u_j \geq b_i$, $i = 1, \ldots, m$. Let us assume that this is the case and let us also assume that the matrix $A = (a_{ij})$ is in standard greedy form. When the setup costs c_j are not all equal we know of no procedure solving the problem in polynomial time, even on tree networks. In the case of equal setup costs we may assume that $c_j = 1$, $j = 1, \ldots, n$. Let us define $y_j = u_j - x_j$, $j = 1, \ldots, n$. The multiple covering problem in case of equal setup costs can be formulated as

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{n} u_j - \sum_{j=1}^{n} y_j \\ &\text{subject to} && \sum_{j=1}^{n} a_{ij} y_j \leq \sum_{j=1}^{n} a_{ij} u_j - b_i\,, \qquad i = 1, \ldots, m \\ &&& 0 \leq y_j \leq u_j,\ y_j \text{ integer}\,, \qquad j = 1, \ldots, n\,, \end{aligned} \tag{6.2.21}$$

or equivalently

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^{n} y_j \\ &\text{subject to} && \sum_{j=1}^{n} a_{ij} y_j \leq \sum_{j=1}^{n} a_{ij} u_j - b_i\,, \qquad i = 1, \ldots, m\,, \\ &&& 0 \leq y_j \leq u_j\,, \qquad y_j \text{ integer}\,, \qquad j = 1, \ldots, n\,. \end{aligned} \tag{6.2.22}$$

Since A, and therefore also its transpose, is in standard greedy form, (6.2.22) is equivalent to (6.2.15). We have already proved that an optimal solution of (6.2.15) is given by (6.2.19). Furthermore, it follows that if c_j $(j=1,\ldots,n)$ and p_i $(i=1,\ldots,m)$ in (6.2.15) are integer, then the constructed solution is also integer. Assuming b_i $(i=1,\ldots,m)$ and u_j $(j=1,\ldots,n)$ are integer, the following algorithm will solve the multiple covering problem. Define the variables x_j in (6.2.20), for increasing index of j, to be as small as possible; that is, use the following recursive relation:

$$x_j = \max\left\{0, \max_{i\,:\,a_{ij}=1}\left\{b_i - \sum_{k=j+1}^{n} a_{ik}u_k - \sum_{k=1}^{j-1} a_{ik}x_k\right\}\right\}. \qquad (6.2.23)$$

Let us interpret (6.2.23). $\Sigma_{k=1}^{j-1} a_{ik}x_k$ is the number of times client i is "covered" (served) by facilities that have already been established. $\Sigma_{k=j+1}^{n} a_{ik}u_k$ is an upper bound on the number of times facilities, which might be setup at sites $\{v_{j+1},\ldots,v_n\}$, can cover this client. Thus, if the sum of the two is less than b_i, then the difference should be accounted for by x_j. Note that if $a_{ij}=1$, then $b_i - \Sigma_{k=j+1}^{n} a_{ik}u_k - \Sigma_{k=1}^{j-1} a_{ik}x_k \leq u_j$ since $b_i \leq \Sigma_{k=1}^{n} a_{ik}u_k$ by assumption.

6.2.2. Transformation to Standard Greedy Form

To be able to apply the algorithm given in the previous section to the covering problem (6.2.2) and the multiple covering problem (6.2.21) for both the client and facility constrained versions on trees, we need to show how to transform the intersection matrix of nodes versus neighborhood subtrees (of a tree) into standard greedy form. This transformation is based on the following theorem (see Kolen, 1983) which we shall prove later in this section.

Theorem 6.2. *Let $N(x_1, r_1)$ and $N(x_2, r_2)$ be two neighborhood subtrees containing an endpoint t_1 of a longest path in the tree. Then $N(x_1, r_1) \subseteq N(x_2, r_2)$ or $N(x_2, r_2) \subseteq N(x_1, r_1)$.*

Theorem 6.2 is no longer true if we replace the endpoint t_1 of a longest path by an arbitrary "tip node," that is, by a node which is adjacent to exactly one other node.

The result of Theorem 6.2 constitutes a generalization of the following property of intervals. If $[a, b]$ is some interval, and I_1 and I_2 are subintervals containing point a, then $I_1 \subseteq I_2$ or $I_2 \subseteq I_1$.

Before proving this theorem let us examine its implications. Consider the intersection matrix of m nodes versus n neighborhood subtrees of a tree T and suppose that we have ordered its rows such that the first row corresponds to a vertex, say t_1, which is an endpoint of a longest path in T. Then we know from Theorem 6.2 that all columns having a 1 in row 1 (that is all neighborhood subtrees containing t_1), can be totally ordered by inclusion,

that is, for any two of them one is contained in the other. Since t_1 is a tip node of the tree we can remove it and the unique arc containing it from the tree T. Let T_1 be the resulting tree. It is easy to see that the intersection of a neighborhood subtree $N(x, r)$ of T with T_1 is a neighborhood subtree of T_1. Thus, there is a $y \in T_1$ and a radius s such that $N(y, s) = N(x, r) \cap T_1$ (see Exercise 6.1). Using this we can repeat the above argument. In general, let t_{i+1} be the node corresponding to row $i+1$, were t_{i+1} is defined to be an endpoint of a longest path in T_i, and where T_i is the subtree obtained by deleting t_i and the unique arc of T_{i-1} containing t_i from T_{i-1}, $i = 1, \ldots, m-1$. (T_0 is defined to be the original tree T.) Using Theorem 6.2 and Exercise 6.1 we have established that the intersection matrix of nodes (rows) versus neighborhood subtrees (columns) has the *nest ordering property for columns* whenever the rows of the intersection matrix are defined recursively above.

Property 6.4. An $m \times n$ $(0, 1)$ matrix has the *nest ordering property for columns* if for $i = 1, \ldots, m$ the following holds: all columns containing a 1 in row i can be totally ordered by inclusion when they are restricted to rows with index greater than or equal to i (see the matrix of Example 6.1 for an illustration).

To find the permutation of the m nodes (rows) corresponding to the nest ordering property for columns we use the following observation (see Exercise 6.2).

Property 6.5. Let v be a given node of a tree T. Then a node of T which is at a largest distance from v is an endpoint of a longest path in T (see the tree in Figure 6.3 for an example).

Thus the nodes $t_1, \ldots, t_m$, which are recursively defined above, and correspond to the permutation of the rows can be obtained as follows. Find the distances from some given node v to all m nodes of the tree (this takes $O(m)$ time). Sort these distances and suppose that $d(v, v_{i1}) \geq d(v, v_{i2}) \geq \cdots \geq d(v, v_{im})$. Then set $t_1 = v_{i1}$, $t_2 = v_{i2}, \ldots, t_m = v_{im} (= v)$. The effort involved in computing and sorting these distances is $O(m \log m)$. Therefore the permutation of the rows yielding a nest ordering property for columns is obtained in $O(m \log m)$ time.

Given this permutation of the rows, one more step is needed now to transform the matrix into the standard greedy form. For $(0, 1)$ vectors x and y, of the same dimension, we say that x is *lexical* larger than y if x and y are different and in the last coordinate in which they differ x has a 1 and y has a 0. (Note that this is not the same as lexicographically larger; in that case the first coordinate in which the vectors differ would be important.) We say that a set of vectors $x^1, \ldots, x^k$ is in *nondecreasing lexical order* if $x^i = x^{i+1}$ or x^{i+1} is lexical larger than x^i, $i = 1, \ldots, k-1$.

Lemma 6.1. *If a* $(0, 1)$ *matrix has the nest ordering property for columns and the columns are ordered in a lexical nondecreasing order, then the matrix is in standard greedy form.*

Proof. Suppose we are given a 2×2 submatrix, as defined by the rows $i(1)$, $i(2)$ (with $i(1) < i(2)$) and columns $j(1)$, $j(2)$ (with $j(1) < j(2)$):

$$\begin{array}{c} \\ i(1) \\ i(2) \\ i(3) \end{array} \begin{array}{c} \begin{array}{cc} j(1) & j(2) \end{array} \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}.$$

Since columns $j(1)$ and $j(2)$ differ, $j(2)$ is lexical larger than $j(1)$. Therefore there exists a row $i(3)$ (with $i(3) > i(2)$) such that column $j(1)$ has a 0 and column $j(2)$ has a 1 in this row. But this contradicts the nest ordering property for the columns which implies that column $j(1)$ is contained in column $j(2)$ or vice versa, whenever we restrict these columns to rows with an index of at least $i(1)$. □

Ordering n $(0, 1)$ vectors of dimension m in a lexical ordering can be done in $O(mn)$ time using a radix sort procedure (see Aho, Hopcroft, and Ullman 1974). We conclude that the transformation of the intersection matrix of the nodes of a tree (rows) versus n neighborhood subtrees (columns) into standard greedy form can be achieved in $O(nm + m \log m)$ time.

The following example illustrates the above concepts.

Example 6.2. Consider the tree shown in Figure 6.1. Let us root the tree at $v = v_3$ and calculate the distances to all other nodes and order them in nondecreasing order, say $d(t_1, v) \geq d(t_2, v) \geq \cdots \geq d(t_7, v)$ with $t_7 = v$. Then if we take t_i as the node corresponding to row i of the matrix, $i = 1, \ldots, 7$, this will give the desired permutation.

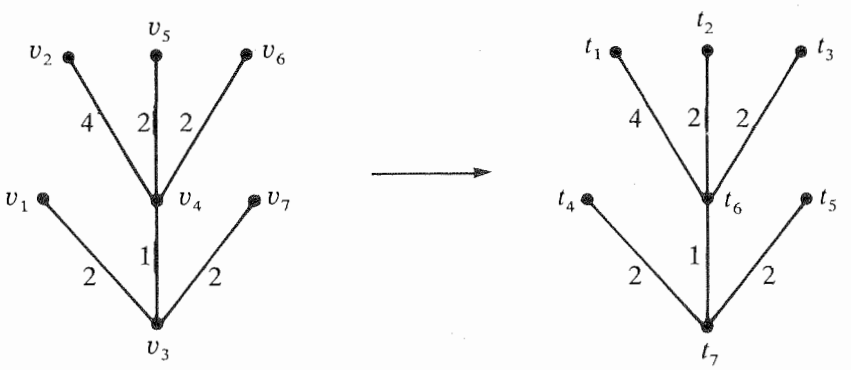

Figure 6.1. Tree of Example 6.2.

Let us consider the following neighborhood subtrees $N_5 = N(v_5, 4)$, $N_2 = N(v_2, 6)$, $N_1 = N(v_1, 3)$, $N_3 = N(v_3, 2)$, $N_7 = N(v_7, 1)$, $N_4 = N(v_4, 2)$, $N_6 = N(v_6, 4)$. A lexical nondecreasing ordering is given by the matrix.

$$
\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} N_7 & N_4 & N_5 & N_6 & N_2 & N_1 & N_3 \end{array} \\
\left[\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]
\end{array}
$$

(The transpose of the first six rows of the matrix was used in Example 6.1) ○

Let us return to Theorem 6.2.

Proof of Theorem 6.2. Let $P[t_1, t_2]$ be a longest path in the tree T. $N(x_1, r_1)$ and $N(x_2, r_2)$ both contain t_1. Define T' to be the minimal subtree of T which contains $N(x_1, r_1) \cup N(x_2, r_2)$. Clearly, t_1 is in T'. Since t_1 is an endpoint of a longest path in T, it is also an endpoint of some longest path of every subtree of T containing t_1. Thus, let t_3 be some point in T' such that $P[t_1, t_3]$ is a longest path in T'. Due to the minimalilty of T' we have $t_3 \in N(x_1, r_1) \cup N(x_2, r_2)$. Suppose without loss of generality that $t_3 \in N(x_1, r_1)$. Then, $N(x_1, r_1)$ contains $P[t_1, t_3]$. Therefore $N(x_1, r_1) = T'$ (see Exercise 6.4), and $N(x_2, r_2) \subseteq T' = N(x_1, r_1)$. This completes the proof. □

It was shown above that the intersection matrix of the m nodes of a tree (rows) versus n neighborhood subtrees (columns) can be transformed into standard greedy form in $O(nm + m \log m)$ time. Consider the case where the $m \times n$ matrix corresponds to the intersection of m neighborhood subtrees (rows) versus some subset of n (out of the m) nodes of the tree (columns) (each neighborhood is centered at some node of the tree). In this case the standard greedy form can be achieved in $O(nm)$ time only. To see that, consider the transpose of this matrix, which is $n \times m$ and transform it to standard greedy form.

First we permute the n rows (nodes) of this transposed matrix. The first row of the permuted matrix is obtained as follows. Iteratively remove all tips of the tree (with their adjacent arcs) which are not in the above subset of n nodes. The remaining tree has only tips belonging to this subset of n nodes. Then find a node which is an endpoint of a longest path of the remaining tree. This node will correspond to the first row. To find the second row, remove this node from the tree, and repeat the above process. Continuing this procedure for n iterations, we obtain the permutation of the rows.

From Property 6.5 it follows that each iteration can be performed in

$O(m)$ time. Therefore the permutation of the rows consumes $O(mn)$ time. The lexical ordering of the columns can also be done in $O(mn)$ time by radix sorting as discussed above. Thus, the standard greedy form is obtained in this case in total time of $O(mn)$.

Summarizing, we conclude that the results in Sections 6.2.1 and 6.2.2 imply the polynomial solvability of both the client and the facility constrained covering problems, as well as the UFL problem, on tree networks. For example, the methods in these sections solve the client constrained problem and the UFL problem on trees in $O(mn)$ and $O(mn^2)$ times, respectively. (Recall that the UFL has been formulated in (6.2.12) as a covering problem with an $(m \cdot n) \times n$ matrix in standard greedy form.)

Polynomial algorithms for these problems can also be constructed using dynamic programming principles. The dynamic programming approach utilizes the natural partial ordering of the nodes induced by rooting a tree at an arbitrary node (see Megiddo, Zemel, and Hakimi, 1983).

6.2.3. The Cost Allocation Problem

It has been shown earlier that both the covering problem and the UFL problem can be formulated as

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{n} c_j x_j + \sum_{\omega \in M_1} p_\omega z_\omega \\ &\text{subject to} && \sum_{j=1}^{n} a_{\omega j} x_j + z_\omega \geq 1 , \qquad \omega \in M_1 , \\ & && x_j \in \{0, 1\} , \qquad j = 1, \ldots, n , \\ & && z_\omega \in \{0, 1\} , \qquad \omega \in M_1 . \end{aligned} \tag{6.2.24}$$

In both cases the variables $x_1, \ldots, x_n$ correspond, respectively, to the n potential facility sites. In the case of the covering problem with m clients M_1 is defined by $M_1 = \{1, \ldots, m\}$ and each index in M_1 is associated with a different client (see the definition of the covering problem). In the case of the UFL problem (see (6.2.12)), $|M_1| = m \cdot n$, and each index $\omega \in M_1$ corresponds to a pair (i, k) where i is a client and k is a facility site. In particular, each client i is associated with a subset of n indices in M_1. If we denote this subset by I_i, $i = 1, \ldots, m$, then $I_1, \ldots, I_m$ constitute a partition of M_1. Thus, in both the covering problem and the UFL problem we have a partition of the rows of the matrix $A = (a_{\omega j})$ into m subsets $I_1, \ldots, I_m$, where I_i, $i = 1, \ldots, m$, is associated with client i. For the covering problem $|I_i| = 1$, while for the UFL problem $|I_i| = n$, $i = 1, \ldots, m$.

We have shown how to solve the location models defined by (6.2.24) in the case where the matrix $A = (a_{\omega j})$ is in standard greedy form. However, we have not addressed two important issues related to these location models. First there exists the question of cooperation between the clients. Is

there an incentive for them to cooperate? Suppose that each one of two subsets of clients operates on its own and establishes facilities serving its members only. Since we have assumed that there are no capacity constraints on the facilities, it is clear that if these two subsets of clients (coalitions) unite, their total cost will not increase. Therefore, viewing only the total cost of all clients, there exists an incentive for all of them to cooperate and act as a grand coalition.

The second question that arises is on the allocation of the total cost. Is there an "acceptable" allocation, an allocation such that no group of clients will have an incentive to not cooperate by splitting from the grand coalition and acting on its own? In game theory terminology such an allocation is called a *core allocation*. Namely, a core allocation of the total cost is such that no coalition can pay less than its part in this allocation if it establishes facilities that will serve its members only. The total cost incurred by a coalition of clients $I \subseteq \{1, \ldots, m\}$, if they want to act alone, is given by $v(I)$, where $v(I)$ is the optimum value of (6.2.25).

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j + \sum_{i \in I} \sum_{\omega \in I_i} p_\omega z_\omega \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{\omega j} x_j + z_\omega \geq 1 , \qquad \omega \in \bigcup \{I_i : i \in I\} , \\ & x_j \in \{0, 1\} , \qquad j = 1, \ldots, n , \\ & z_\omega \in \{0, 1\} , \qquad \omega \in \bigcup \{I_i : i \in I\} \end{aligned} \tag{6.2.25}$$

where I_i is the set of constraints corresponding to client i, $i = 1, \ldots, m$.

Let us define the core of the covering problem by

$$\text{core}\,(\{1, \ldots, m\}) = \left\{ y \in R^m : \sum_{i=1}^{m} y_i = v(\{1, \ldots, m\}) \text{ and } \sum_{i \in I} y_i \leq v(I) \text{ for all } I \subseteq \{1, \ldots, m\} \right\} . \tag{6.2.26}$$

If $y \in \text{core}\,(\{1, \ldots, m\})$, then it is called a core allocation and y_i denotes the part of the total cost paid by client i, $i = 1, \ldots, m$. It is clear that no coalition of clients would do better by breaking the cooperation between all clients if a core allocation is used to split the total cost. Thus a core allocation possesses a desirable stability property which seems to be necessary for an allocation to be acceptable by all the clients.

We next give an example of a covering problem for which the core is empty. The example is defined on a network which is a cycle. This motivates us to look at trees; we will show that for covering problems on trees the core is nonempty.

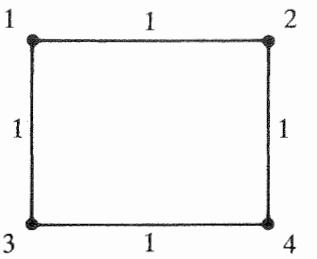

Figure 6.2. Network for Example 6.3.

Example 6.3. Consider the network shown in Figure 6.2. Each arc has length one. We consider the UFL problem with setup cost equal to 1.5 for all nodes and transportation costs which are equal to the respective distances from clients to their closest established facilities. It is easily verified that an optimal solution to the problem is given by establishing facilities at 1 and 2 and letting 3 and 4 be served by 1 and 2, respectively. The total cost is 5. If the UFL problem is restricted to any subset of three clients the optimum value is 3.5. Therefore every vector y in the core must satisfy:

$$\begin{aligned} y_1 + y_2 + y_3 + y_4 &= 5 \\ y_1 + y_2 + y_3 &\leq 3.5 \\ y_1 + y_2 + y_4 &\leq 3.5 \\ y_1 + y_3 + y_4 &\leq 3.5 \\ y_2 + y_3 + y_4 &\leq 3.5\,. \end{aligned}$$

Adding the four inequalities together yields $3(y_1 + y_2 + y_3 + y_4) \leq 14$. But the latter contradicts $3(y_1 + y_2 + y_3 + y_4) = 15$ which is the first equation. Therefore we conclude that the core is empty. ○

Having observed that cycles may yield an empty core, we turn to tree networks and prove that there the core is always nonempty. If u is the optimal solution of the dual of the LP-relaxation of (6.2.24), then we will show that $\hat{y}_i = \Sigma_{\omega \in I_i} u_\omega$, $i = 1, \ldots, m$, will generate a core allocation. Remember that I_i is the set of constraints corresponding to client i in (6.2.24).

Theorem 6.3. *If the problem* (6.2.24) *arises from a tree location problem, that is, the matrix* $(a_{\omega j})$ *is in standard greedy form, and* u *is an optimal solution to the dual of the LP-relaxation of* (6.2.24), *then* $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$ *defined by* $\hat{y}_i = \Sigma_{\omega \in I_i} u_\omega$ *is in the core defined by* (6.2.26).

Proof. It is shown in Section 6.2.1 that the optimal objective value of a covering problem, defined by a matrix in standard greedy form, is equal to

the optimal value of its linear programming relaxation. Thus, duality yields

$$v(\{1, \ldots, m\}) = \sum_{\omega \in M_1} u_\omega = \sum_{i=1}^{m} \sum_{\omega \in I_i} u_\omega = \sum_{i=1}^{m} \hat{y}_i .$$

Therefore to prove that $\hat{y}$ is in the core it suffices to show that

$$\sum_{i \in I} \sum_{\omega \in I_i} u_\omega \leq v(I) \qquad \text{for all} \qquad I \subseteq \{1, \ldots, m\} .$$

Let $I \subseteq \{1, \ldots, m\}$ and consider the dual problem of the LP-relaxation of (6.2.25). Again by the results of Section 6.2.1 we know that the optimal solution to the dual of this problem is equal to $v(I)$. If we prove that u_ω, $\omega \in I_i$, $i \in I$ is a feasible solution to this dual problem, then the result follows from the weak duality property. Of course, we know that $0 \leq u_\omega \leq p_\omega$, $\omega \in M_1$, so that it suffices to prove that $\Sigma_{i \in I} \Sigma_{\omega \in I_i} u_\omega a_{\omega j} \leq c_j$ for $j = 1, \ldots, n$, but this is trivial since the left-hand side is less than or equal to $\Sigma_{\omega \in M_1} u_\omega a_{\omega j}$ for which we know that it is less than or equal to c_j, $j = 1, \ldots, n$. □

We refer the reader to a paper by Tamir (1980) for a treatment on a more general cost-allocation game on trees.

6.3. CENTER PROBLEMS

In this section we discuss the budget constrained center and round-trip center problems on trees. We assume that clients are located only at the nodes of the tree. Each node is identified as a client. At the end of this section we also refer to other cases. We consider the case where facilities can be established only at the nodes as well as the infinite case where we may establish a facility anywhere along the tree.

The round-trip distance occurs, for example, in the context of a transportation problem in which a company has to execute a number of jobs. Job i consists of picking up goods at node a_i and delivering them at node b_i. The distance a vehicle located at a depot x has to travel in order to execute job i and return to its depot is given by the *round-trip distance* $d(x, a_i) + d(a_i, b_i) + d(b_i, x)$. If we let $D(P_i, x)$ denote the distance from x to the closest point on P_i, the shortest path connecting a_i and b_i, then the round-trip distance is given by $2[d(a_i, b_i) + D(P_i, x)]$ (see Figure 6.3). If X is a closed set of points on the network, then $D(P_i, X) := \min_{x \in X} \{D(P_i, x)\}$.

For ease of exposition we will assume, while defining the center problems, that the transportation cost between client i and the closest established facility x is $d(v_i, x)$. Similarly, the transportation cost between a pair of

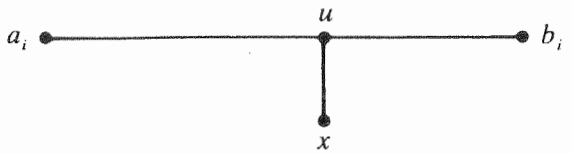

Figure 6.3. Illustrating that $D(P_i, x) = d(u, x)$, and round-trip distance $= 2[d(a_i, b_i) + D(P_i, x)]$.

clients a_i, b_i and x is $D(P_i, x)$. The case where these costs are nondecreasing functions of the distances can be treated similarly. Note that the round-trip distance $2[d(a_i, b_i) + D(P_i, x)]$ is a nondecreasing function of $D(P_i, x)$.

Let us start by considering the budget constrained center problem, where facilities can be established at the nodes only. The setup cost of establishing a facility at v_j is c_j, $j = 1, \ldots, m$. (Here we assume that all nodes are potential sites.) The *budget constrained center problem* is to minimize the maximum of the distances of the clients to their respective nearest facilities, subject to a budget constraint on the total setup cost. It is formulated as follows:

$$\text{minimize} \quad z \tag{6.3.1}$$

$$\text{subject to} \quad \min_{v_j \in S} \{d(v_i, v_j)\} \leq z\,, \qquad i = 1, \ldots, m\,, \tag{6.3.2}$$

$$\sum_{j:\, v_j \in S} c_j \leq B\,, \qquad S \subseteq V \tag{6.3.3}$$

where S denotes the set of established facilities. Notice that z can only take on values belonging to the set R, where

$$R = \{d(v_i, v_j)\colon i, j = 1, \ldots, m\}\,. \tag{6.3.4}$$

It is easy to prove the following claim. The optimum value of (6.3.1) is the smallest value z, say z^*, in the set R for which the client constrained covering problem (6.2.2), with radius $r_i = z^*$, and penalty $p_i = \infty$, $i = 1, \ldots, m$, has a total setup cost which does not exceed B. Hence, we can solve this budget constrained center problem by solving a sequence of covering problems. Using a binary search over the above set R, the optimal solution will be found after solving $O(\log m)$ covering problems.

In the case when all setup costs are equal, the budget constraint simply specifies the number, say p, of facilities that may be established. In this case the problem is well recognized as *the p-center problem*. We will defer our discussion of this case, and treat it together with the respective case of the round-trip center problem. The general p-center problem is extensively discussed, in another perspective, in Chapter 7.

In contrast with the problem (6.3.1)–(6.3.3), the budget constrained round-trip center problem is $\mathcal{NP}$-hard when facilities, which are restricted to

be located at the nodes, may differ in their setup costs. Recall that in this problem we are given q pairs of clients $\{a_i, b_i\}$, $i = 1, \ldots, q$, and the objective is to minimize the maximum of the distances of the pairs to their respective nearest facilities, subject to a budget constraint. Formally, the *budget constrained round-trip center problem* is:

$$\text{minimize} \quad z \tag{6.3.5}$$

$$\text{subject to} \quad \min_{v_j \in S} \{D(P_i, v_j)\} \leq z\,, \qquad i = 1, \ldots, q \tag{6.3.6}$$

$$\sum_{j:\, v_j \in S} c_j \leq B\,, \qquad S \subseteq V \tag{6.3.7}$$

where S denotes the set of established facilities (depots).

The $\mathcal{NP}$-hardness is proven by reducing the *node cover problem*, a known $\mathcal{NP}$-complete problem, to this problem. Given an undirected graph $G = (\bar{V}, \bar{E})$ with m nodes and a number $k \leq m$, the node cover problem is to determine whether there exist a subset V' of $\bar{V}$ of at most k nodes, such that each arc contains at least one node in V'.

Consider the tree T in Figure 6.4. Suppose that $\bar{V} = \{u_1, \ldots, u_m\}$. Then $\{v_i, v_j\}$ is defined to be a pair of clients in T if and only if $[u_i, u_j]$ is an arc of G. The setup costs associated with the nodes of T are as follows: $c_i = 1$, $i = 1, \ldots, m$ and $c_{m+1} = m + 1$. Taking the budget constraint B to be k, it is then an easy task to verify that G has a node cover of cardinality k if and only if the solution to the budget constrained round-trip problem is equal to zero. We conclude that the budget constrained round-trip center problem on trees is $\mathcal{NP}$-hard, even for the case where all but one of the setup costs are equal. (Notice that the problem is in fact $\mathcal{NP}$-complete when posed as a recognition problem.) We will show later that if all setup costs are equal then this problem is polynomially solvable.

The above reduction leading to the $\mathcal{NP}$-hardness result is, in fact, valid for a larger class of objective functions. We may consider any nondecreasing objective function of the distances $D(P_1, X), \ldots, D(P_q, X)$, $X \subseteq V$, which is equal to zero if and only if $D(P_i, X) = 0$, $i = 1, \ldots, q$. For example, the median version of the problem, where we try to minimize $\Sigma_{i=1}^{q} D(P_i, X)$ over all feasible subsets X of cardinality p, is also $\mathcal{NP}$-hard when the set of potential location sites consists of all the nodes of the tree but one.

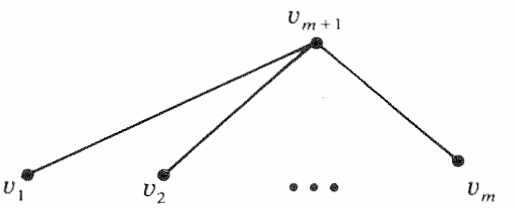

Figure 6.4. The tree of the reduction.

The difference in complexity between the budget constrained center problem and its round-trip counterpart is due to the fact that $T_i = \{y \in T: d(y, v_i) \leq r\}$ is a neighborhood subtree, while, in general, this does not apply to $S_i = \{y \in T: D(P_i, y) \leq r\}$. (Note that P_i is a path in the tree and not a node.)

Let us now introduce some theory to exhibit the polynomial solvability of the center and round-trip center problems, for the case where facilities are allowed to be located anywhere along the tree, with a constant setup cost.

We will assume that the tree is rooted at some arbitrary node, say v_m. For each node v_j define v_k to be a *child* of v_j if v_k is adjacent to v_j and v_j is on the shortest path from v_k to v_m. Note v_j is also called the *parent* of v_k. We also assume without loss of generality that the nodes are indexed in such a way that children have a smaller number than their parent. See Figure 6.5 for an illustration of a tree with 7 nodes rooted at v_7. The children of v_6 are v_1, v_2, and v_3.

Let $T_1, \ldots, T_n$ be subtrees of T. Without loss of generality we may assume that they are induced subgraphs, that is, each tip of a subtree belongs to the node set of T. (If this is not the case the tips are augmented to the node set.) With each subtree T_i we associate a node v_{ri}, called *the root of the subtree* T_i, which is the node with the largest index in T_i. This means that the root of T_i is on the shortest path connecting the root of T with any point in T_i. We then have the following observation.

Property 6.6. Given subtrees T_i and T_j with roots v_{ri} and v_{rj}, respectively, suppose that $ri \leq rj$. Then

$$T_i \cap T_j \neq \varphi \qquad \text{iff} \qquad v_{ri} \in T_j .$$

From this we can obtain the Helly property for trees.

Property 6.7 (Helly Property). Given subtrees $T_1, \ldots, T_n$ such that $T_i \cap T_j$ is nonempty for $i \neq j$, then $\cap_{i=1}^{n} T_i$ is nonempty.

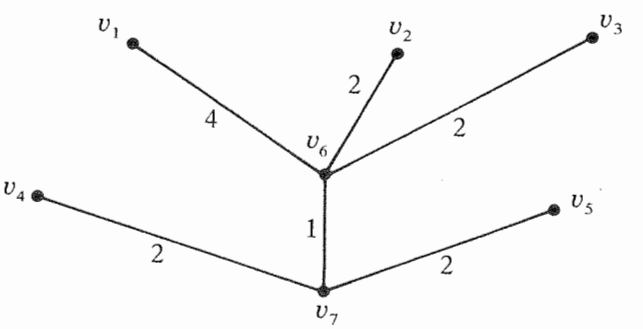

Figure 6.5. Illustration of node indexing.

Proof. Without loss of generality assume $r1 \leq r2 \cdots \leq rn$. Since $T_1 \cap T_i$ is nonempty for $i = 1, \ldots, n$, we have $v_{r1} \in T_i$, $i = 1, \ldots, n$. □

Remark 6.1. It follows from the above proof that when $T_1, \ldots, T_n$ are induced subtrees whose intersection is nonempty, then this intersection contains a node.

The *intersection graph* corresponding to the subtrees $T_1, \ldots, T_n$ is the graph $G = (U, \bar{E})$, with $U = \{u_1, \ldots, u_n\}$ and $[u_i, u_j] \in \bar{E}$ if and only if $i \neq j$ and $T_i \cap T_j$ is nonempty.

A *chordal graph* is an undirected graph with the property that every simple cycle of length at least four contains a *chord*, that is, an arc connecting two nodes which are not adjacent in the cycle. The following theorem states that the intersection graph of subtrees of a tree is a chordal graph. The converse of this theorem is also true, so that every chordal graph is representable as the intersection graph of subtrees of some tree. This one-to-one correspondence between chordal graphs and intersection graphs of subtrees of a tree has been established by Walter (1972, 1978), Buneman (1974), and Gavril (1974). We will present a different and simpler proof for the chordality of the above intersection graph.

Theorem 6.4. *The intersection graph $G = (U, \bar{E})$ of the set of subtrees $T_1, \ldots, T_n$ of a given tree T is chordal.*

Proof. Without loss of generality we assume $r1 \leq r2 \leq, \ldots, \leq rn$. Consider a simple cycle of length at least four and suppose that u_i, the node of G corresponding to T_i, has the smallest index among all nodes of the cycle. Also, let $[u_i, u_j]$ and $[u_i, u_k]$, $k \neq j$, be two arcs in that cycle. Since we have $v_{ri} \in T_j \cap T_k$ (Property 6.6), $[v_j, v_k]$ is a chord of the cycle. □

Remark 6.2. Taking a closer look at the proof of Theorem 6.4 we see that we have actually proved the following. There exists an ordering of the indices of the nodes of the intersection graph, determined by the roots of the subtrees, such that all nodes u_j in G which are adjacent to u_i and have a larger index than u_i form a *clique* in G; that is, a set of nodes which are pairwise adjacent. An ordering with this property is called a *perfect elimination scheme* (see Example 6.4).

One can prove that there exists a perfect elimination scheme for a graph if and only if the graph is chordal (see Golumbic, 1980).

An *independent set* of a graph is a set of nodes such that no two of them are adjacent. A *clique cover* is a set of cliques with the property that each node of the graph is contained in at least one clique. Since each node of an independent set must be in a different clique of the cover we have a weak duality result. The cardinality of any independent set is less than or equal to the cardinality of any clique cover.

The existence of the above ordering for the intersection graph of subtrees of a tree enables us to prove a strong duality result for chordal graphs.

Theorem 6.5. *Let G be an intersection graph of the set of subtrees $T_1, \ldots, T_n$ of a given tree T. Then the maximum cardinality of an independent set in G is equal to the minimum cardinality of a clique cover.*

Proof. We shall construct an independent set and a clique cover of the same cardinality. Consider the perfect elimination scheme induced on the nodes of G by the roots of the subtrees $T_1, \ldots, T_n$. Suppose that u_i is a node of G which is smallest with respect to the ordering. Add this node to the current independent set. From Remark 6.2 we know that all nodes adjacent to u_i together with u_i form a clique in G. Add this clique to the current set of cliques. (Initially, both the independent set and the set of cliques are empty.) Remove the subgraph induced by u_i and its adjacent nodes from G. The remaining graph will still be an intersection graph of subtrees. We will repeat this procedure for the remaining graph, and this will produce the desired clique cover and independent set. □

Example 6.4. Consider the tree shown in Figure 6.5. The subtrees will be given by their nodes, $T_1 = \{v_1\}$, $T_2 = \{v_1, v_2, v_6\}$, $T_3 = \{v_1, v_3, v_6\}$, $T_4 = \{v_4, v_7\}$, $T_5 = \{v_5, v_7\}$, and $T_6 = \{v_6, v_7\}$. We obtain $r1 = 1$, $r2 = 6$, $r3 = 6$, $r4 = 7$, $r5 = 7$, and $r6 = 7$. The intersection graph is given in Figure 6.6, where u_i corresponds to T_i.

It is easily observed that the ordering of the perfect elimination scheme corresponds to the indices of the nodes, u_i, $i = 1, \ldots, 7$. By the procedure described in the above proof $\{u_1, u_4\}$ is a maximum independent set, while $\{u_1, u_2, u_3\}$ and $\{u_4, u_5, u_6\}$ is a minimum clique cover. ○

Let us demonstrate how the duality result of Theorem 6.5 can be used to solve the p-center and round-trip p-center problems, when facilities can be established anywhere on the tree. It will be shown that both problems are solvable by a sequence of covering problems.

The covering problem for the p-center problem is

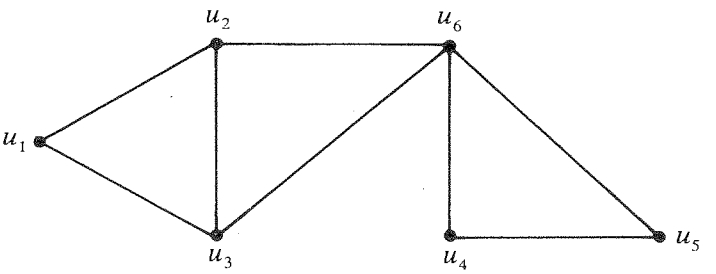

Figure 6.6. Intersection graph for Example 6.5.

$$\begin{aligned} &\text{minimize} && |X| \\ &\text{subject to} && D(v_i, X) \le r\,, \qquad i = 1, \ldots, m\,, \\ & && X \subseteq T \end{aligned} \tag{6.3.8}$$

where $D(v_i, X) = \min_{x \in X} \{d(v_i, X)\}$. (Since X is finite, $|X| \le m$, $D(v_i, X)$ is well defined.)

For the round-trip p-center problem we obtain the following covering problem

$$\begin{aligned} &\text{minimize} && |X| \\ &\text{subject to} && D(P_i, X) \le r\,, \qquad i = 1, \ldots, q\,, \\ & && X \subseteq T \end{aligned} \tag{6.3.9}$$

where $D(P_i, X) = \min_{x \in X} \{D(P_i, x)\}$.

Focussing first on (6.3.8), define the subtrees $T_i = \{y \in T: d(v_i, y) \le r\}$, $i = 1, \ldots, m$. It is easily verified that

$$T_i \cap T_j \quad \text{is nonempty iff} \quad d(v_i, v_j) \le 2r\,, \qquad i, j = 1, \ldots, m\,. \tag{6.3.10}$$

If $G = (U, \bar{E})$, $U = \{u_1, \ldots, u_m\}$ is the intersection graph of $T_1, \ldots, T_m$, then $[u_i, u_j] \in \bar{E}$ if and only if $i \ne j$ and $d(v_i, v_j) \le 2r$. If two clients v_i, v_j can be served by the same facility, then $T_i \cap T_j$ is nonempty. Therefore, clients which can be served by the same facility form a clique in G. Conversely, any two subtrees corresponding to nodes in the same clique are pairwise nondisjoint. According to the Helly property (Property 6.7), there exists a point of the tree which is contained in all subtrees corresponding to the same clique. This means that a facility located in such a point can serve all clients corresponding to this clique, that is, the distance of these clients to the facility is at most r. This proves that there is a one-to-one correspondence between the cliques of G and the subsets of the clients that can be served by the same facility. Therefore, finding the minimum number of facilities which can serve all clients within a radius r, is equivalent to finding a minimum clique cover in G.

By the duality result of Theorem 6.5 and (6.3.10), the minimum number of facilities is equal to the maximum number of clients for which the mutual distance is greater than $2r$. The latter problem is formulated as

$$\begin{aligned} &\text{maximize} && |I| \\ &\text{subject to} && d(v_i, v_j) > 2r\,, \qquad i, j \in I\,, \qquad i \ne j\,, \\ & && I \subseteq \{1, 2, \ldots, m\}\,. \end{aligned} \tag{6.3.11}$$

For the round-trip center problem we define $T_i = \{y \in T: D(P_i, y) \leq r\}$, $i = 1, \ldots, q$. We leave it to the reader to show that

$$T_i \cap T_j \text{ is nonempty iff } \gamma_{ij} \leq 4r \tag{6.3.12}$$

where

$$\gamma_{ij} = d(a_i, a_j) + d(b_i, b_j) - d(a_i, b_i) - d(a_j, b_j), \qquad i, j = 1, \ldots, q.$$

Using exactly the same reasoning as for the center problem we can show that the optimal solution to (6.3.9) is equal to the solution of

$$\begin{aligned} &\text{maximize} && |I| \\ &\text{subject to} && \gamma_{ij} > 4r, \quad i, j \in I, \quad i \neq j, \quad I \subseteq \{1, \ldots, q\}. \end{aligned} \tag{6.3.13}$$

The usefulness of these duality results is demonstrated by the next two theorems. They define sets (in fact of polynomial cardinality), that contain the optimal values for the p-center and round-trip p-center problems, respectively.

Theorem 6.6. *Suppose that $p < m$, then*

$$\begin{aligned} &\max_{I \subseteq \{I, \ldots, m\}, |I| = p+1} \left\{ \min_{i, j \in I, i \neq j} \{d(v_i, v_j)/2\} \right\} \\ &\quad = \min_{X \subseteq T, |X| = p} \left\{ \max_{i = 1, \ldots, m} \{D(v_i, X)\} \right\}. \end{aligned} \tag{6.3.14}$$

Proof. It is sufficient to show that for each $r \geq 0$, the right-hand side of (6.3.14) is less than or equal to r if and only if the left-hand side is less than or equal to r. The following equivalence is straightforward.

$$\min_{X \subseteq T, |X| = p} \left\{ \max_{i = 1, \ldots, m} \{D(v_i, X)\} \right\} \leq r$$

if and only if

$$\min \left\{ |X|: \left\{ \max_{i = 1, \ldots, m} \{D(v_i, X)\} \right\} \leq r, X \subseteq T \right\} \leq p. \tag{6.3.15}$$

Using the duality between (6.3.8) and (6.3.11), (6.3.15) is equivalent to

$$\max \{|I|: I \subseteq \{1, \ldots, m\}, d(v_i, v_j) > 2r, i, j \in I, i \neq j\} \leq p, \tag{6.3.16}$$

which, in turn, is immediately observed to be equivalent to

$$\max_{I\subseteq\{1,\ldots,m\},\,|I|=p+1}\left\{\min_{i,\,j\in I,\,i\neq j}\{d(v_i,v_j)/2\}\right\}\leq r\,.\quad\square \tag{6.3.17}$$

Performing the same analysis for the round-trip p-center problem we obtain the following theorem.

Theorem 6.7. *Suppose that $p<q$, then*

$$\max_{I\subseteq\{1,\ldots,q\},\,|I|=p+1}\left\{\min_{i,\,j\in I,\,i\neq j}\gamma_{ij}/4\right\}=\min_{X\subseteq T,\,|X|=p}\left\{\max_{i=1,\ldots,q}\{D(P_i,X)\}\right\}.$$

Theorem 6.6 implies that the optimal solution to the p-center problem is an element in the set

$$R=\{d(v_i,v_j)/2\colon i,j=1,\ldots,m\}\,. \tag{6.3.18}$$

Similarly, Theorem 6.7 implies that the optimal solution to the round-trip p-center problem is an element in

$$R'=\{\gamma_{ij}/4\colon i,j=1,\ldots,q\}\,. \tag{6.3.19}$$

This means that we can solve the two center problems as a sequence of covering problems. For example, the optimal solution to the p-center problem is the smallest element r in R for which the solution to the respective covering problem (6.3.8) is at most p. A similar observation applies to the round-trip p-center problem, when the respective covering problem is (6.3.9). Polynomial algorithms for the covering problems (6.3.8) and (6.3.9) are described by Kariv and Hakimi (1979a) and Kolen (1985), respectively. These algorithms work on the tree networks (see also Slater, 1976). One can also generate the respective intersection graphs and find minimum clique covers on these chordal graphs by the polynomial schemes of Gavril (1972) and Rose, Tarjan, and Lueker (1976).

An efficient implementation of the approach to solve the p-center problem on trees as a sequence of covering problems is presented by Megiddo, Tamir, Zemel, and Chandrasekaran (1981). It is based on an efficient search of the above set R (6.3.18), without explicitly generating it. To illustrate the idea of searching the above set R without explicitly generating its $|R|=O(m^2)$ elements, consider the special case where the tree is a path. In this case the nodes $\{v_1,\ldots,v_m\}$ may be identified as numbers on the real line. Suppose that $v_1\leq v_2\leq\cdots\leq v_m$. Then $R=\{\frac{1}{2}(v_j-v_i)\colon 1\leq i\leq j\leq m\}$. Define

$$R_i=\{(v_j-v_i)/2\colon j=i,i+1,\ldots,m\}\,,\qquad i=1,\ldots,m\,.$$

Then R is the union of the sets R_i, $i = 1, \ldots, m$. The search in R for the optimal solution to the p-center problem consists of $O(\log m)$ steps. In each step we delete parts of the sets R_i, $i = 1, \ldots, m$, which are known not to contain the optimal value. Furthermore, the cardinality of R, the union of the sets R_i, $i = 1, \ldots, m$, will decrease by a factor of at least $1/4$ at each step.

Throughout the search each set R_i, $i = 1, \ldots, m$, will be represented as $R_i = \{(v_i - v_j)/2\colon\ j = a(i),\ a(i)+1, \ldots, b(i)\}$, where $i \le a(i) \le b(i) \le m$, and $a(i)$, $b(i)$ are updated at each step. (Initially $a(i) = i$ and $b(i) = m$). So the elements of R_i are not calculated explicitly but represented by $a(i)$ and $b(i)$. The elimination of elements from R is performed as follows. We first compute the median element, s_i, of each set R_i, $i = 1, \ldots, m$. s_i is given by $(v_k - v_i)/2$, where $k = [\frac{1}{2}(b(i) - a(i) + 1)]$.

Having computed $\{s_1, \ldots, s_m\}$ we associate a weight, w_i, with s_i, $i = 1, \ldots, m$, where $w_i = b(i) - a(i) + 1$, is the number of elements in the current set R_i. Using the method of Aho et al. (1974), we then find in $O(m)$ time, the weighted median, say r, of the set $\{s_1, \ldots, s_m\}$. Next we solve the covering problem (6.3.8) for the value r, using the $O(m)$ procedure of Kariv and Hakimi (1979a). If the optimum of the covering problem is less than or equal to p, then the optimum value of the p-center problem is less than or equal to r. Therefore, if $s_i \ge r$, the elements in R_i which are at least as large as s_i can be deleted from R_i. Similarly, if the optimum to the covering problem is greater than p, the optimum value of the p-center problem is greater than r. Thus, if $s_i \le r$ the elements in R_i which are smaller than or equal to s_i can be discarded.

Since r is a weighted median of the sequence $\{s_1, \ldots, s_m\}$, it is easy to see that we will delete at each step at least $1/4$ of the elements in R, the union of the sets R_i, $i = 1, \ldots, m$. Note that the deletion of elements in a set R_i is done in constant time by increasing $a(i)$ or decreasing $b(i)$, depending on the case. Hence, the total time for one step of this procedure is $O(m)$. Since at each step the cardinality of R is decreased by a factor of at least $1/4$, after $O(\log m)$ steps, R will contain one element, the optimal value to the p-center problem. We conclude that the total complexity of this algorithm to solve the p-center problem on a tree which is a path is $O(m \log m)$.

The p-center problem on a tree with m nodes is solved by Megiddo et al. (1981) in $O(m \log^2 m)$ time. A further improvement of this implementation was achieved by Frederickson and Johnson (1983), who reduced the time bound to $O(m \log m)$. An $O(m)$ algorithm for the case $p = 1$ is given by Handler (1973). A more general case of the above p-center problem, where the transportation costs are arbitrary linear functions of the distance is solved by Megiddo and Tamir (1983) in $O(m \log^2 m \log\log m)$ time.

The set R' in (6.3.19) is the set of elements of the form $\frac{1}{2}D(P_i, P_j)$, $i, j = 1, \ldots, q$, where P_i and P_j are the shortest paths on T connecting the pair of clients a_i, b_i and a_j, b_j, respectively. When the clients are restricted

to the nodes of T (as was assumed above), each distance $D(P_i, P_j)$ is also the distance between some two nodes of the tree. We conclude that R' in (6.3.19) is a subset of R in (6.3.18). Therefore, while looking for the optimal solution to the round-trip p-center problem, we can either search over R in (6.3.18) or R' in (6.3.19), depending on which is more convenient.

We remind the reader that duality results similar to Theorems 6.6 and 6.7 are easily obtained for more general transportation cost functions than the identity function we have chosen here for ease of exposition (see Tansel, Francis, Lowe, and Chen, 1982; Kolen, 1985).

A comment is in order with respect to the round trip p-center problem when facilities can be established only at the nodes of the tree. For this case we showed above that the problem becomes $\mathcal{NP}$-hard even if we exclude only one node from the set of potential sites (the set of nodes). However, we will use the above analysis, on chordal graphs to show that if *all* nodes of the tree are potential locations, then the round-trip p-center problem is polynomially solvable.

First note that in this case the optimal value of the objective is an element in R^2

$$R^2 = \{d(v_i, v_j)\colon\ i, j = 1, \ldots, m\}\ . \tag{6.3.20}$$

Thus, the round-trip p-center problem is polynomially solvable if the following covering problem is polynomially solvable.

$$\begin{aligned} &\text{minimize} && |X| \\ &\text{subject to} && D(P_i, X) \le r\ , \qquad i = 1, \ldots, q\ , \\ &&& X \subseteq V\ . \end{aligned} \tag{6.3.21}$$

Define $S_i = \{y \in T\colon D(P_i, y) \le r\}$, $i = 1, \ldots, q$, and let S_i' be the subtree induced by the nodes of T that belong to S_i as well. We claim that (6.3.21) is equivalent to finding a minimum clique cover on the intersection graph corresponding to the induced subtrees $S_1', \ldots, S_q'$. In order to prove this claim by the above analysis (which was applied to the case where each point on a tree was a potential facility), it suffices to prove the following. If a subcollection $\{S_{i1}', \ldots, S_{it}'\}$ of $\{S_1', \ldots, S_q'\}$ has a nonempty intersection, then it contains a node. Since S_i', $i = 1, \ldots, q$, are induced subtrees the latter follows from Remark 6.1.

Therefore, we conclude that the round-trip p-center problem where facilities can be established at all nodes and only at nodes, at a constant setup cost, is polynomially solvable.

We conclude this section with a discussion on the continuous p-center problem, that is, where each point of the continuum of points on the tree is identified as a client. We consider two cases. The first one is where facilities

can be established at the nodes only, while in the second case there are no restrictions on the potential location sites.

The first case is easily reducible to the case of a finite set of clients. It is shown by Chandrasekaran and Tamir (1982) that the solution to the problem is not affected if we assume that clients exist only at some finite set of points. Specifically, clients are located only at the nodes and at the midpoints of all shortest paths connecting nodes of the tree. Therefore, we can solve the budget constrained version of this problem by augmenting the above finite set of clients to the set of nodes of the tree and solving the resultant discrete p-center problem as before.

Efficient algorithms solving the second case, that is, when facilities can be established anywhere on the tree are given by Chandrasekaran and Daughety (1981), Chandrasekaran and Tamir (1980), Frederickson and Johnson (1983), and Megiddo and Tamir (1983). The most efficient algorithm known is that of Megiddo and Tamir that runs in $O(m \log^3 m)$ time.

A duality result for this continuous version of the p-center problem was obtained by Shier (1977), using a direct inductive proof. We will generalize his duality result by applying a theorem on infinite chordal graphs.

Consider the p-center problem where clients are located only at some subset of points $D \subseteq T$. D may be finite or infinite. Also suppose that facilities can be established anywhere on the tree. We will now define a generalized covering problem as follows.

Let each client $x \in D$ be associated with a positive radius r_x. Find a subset X of T of minimum cardinality, such that for each $x \in D$ there exists $y \in X$ with $d(x, y) \leq r_x$.

For each x define $T_x = \{y \in T: d(x, y) \leq r_x\}$, and consider the intersection graph G of $\{T_x\}$, $x \in D$. G will have a node v_x for each neighborhood subtree T_x, and v_x and v_y will be adjacent if and only if $x \neq y$ and $T_x \cap T_y$ is nonempty, that is if and only if $d(x, y) \leq r_x + r_y$. If D is infinite so is G. Like in the finite case it can be shown that G is an (infinite) chordal graph. The following theorem on infinite chordal graphs (see Hajnal and Súranyi, 1958; Wagon, 1978) is used to prove our duality result.

Theorem 6.8. *Let G be an infinite chordal graph. Let $\theta(G)$ denote the minimum cardinality of a clique cover, and let $\alpha(G)$ denote the maximum cardinality of an independent set. If either $\theta(G)$ or $\alpha(G)$ is finite, then both are finite and equal.*

To ensure the finiteness of $\alpha(G)$ or $\theta(G)$ for the chordal graph associated with our covering problem, we assume that the radii are uniformly bounded from below. There exists an $\varepsilon > 0$ such that $r_x \geq \varepsilon$ for all $x \in D$. It is easily observed that $\theta(G) \leq m + \Delta/2\varepsilon$, where Δ is the sum of the arc lengths of the tree. Since the Helly property is still valid for this case (T_x is connected and closed, $x \in D$), $\theta(G)$ is the solution to the covering problem defined above.

Using Theorem 6.8 we obtain the following corollary for the case $r_x = r > 0$ for all $x \in D$.

Corollary 6.1. *Let D be a subset of T. Then for any $r > 0$,*

$$\min \{|X|: X \subseteq T,\ d(y, X) \leq r \text{ for all } y \in D\}$$
$$= \max \{|Y|: Y \subseteq D,\ d(y_1, y_2) > 2r \text{ for all } y_1, y_2 \in Y,\ y_1 \neq y_2\}.$$

Corollary 6.1 generalizes the duality between (6.3.8) and (6.3.11) as well as the duality result obtained by Shier (1977) for the case $D = T$. When D is assumed to be closed we obtain the following duality result for the p-center problem.

Theorem 6.9. *Let D be a closed subset of T. Then for each positive p, $p < |D|$,*

$$\min_{X \subseteq T,\ |X| = p} \left\{ \max_{y \in D} \{d(y, X)\} \right\} = \max_{Y \subseteq D,\ |Y| = p+1} \left\{ \min_{x, z \in Y,\ x \neq z} \{d(x, z)/2\} \right\}. \tag{6.3.22}$$

Unlike the finite case, if D is infinite, Theorem 6.8 does not suggest a set R that includes r_p, the objective value of the p-center problem (the left-hand side of 6.3.22). This issue is resolved by Chandrasekaran and Tamir (1980) for the case where $D = T$, and by Tamir and Zemel (1982) for a more general case. For example, when $D = T$ it is shown by Chandrasekaran and Tamir (1980) that $r_p = d(x, y)/2k$, where x and y are some tip nodes of T and k is an integer less than or equal to p. Linear time algorithms solving the covering problem for $D = T$, and some generalizations are presented by Chandrasekaran and Tamir (1980) and Tamir and Zemel (1982), respectively.

6.4. TOTALLY BALANCED MATRICES

The covering problem on trees was introduced and solved in Section 6.2. Its formulation is given by (6.2.2).

The algorithm of Section 6.2 to solve the above integer program did not utilize the fact that the matrix A was obtained by row and column permutations from an intersection matrix of neighborhood subtrees versus nodes or vice versa. It relied only upon the information that the matrix $A = (a_{ij})$ was given in standard greedy form, and produced the optimal solution in $O(mn)$ time. This immediately raises the following questions. What is the class of matrices that can be permuted into standard greedy form? Is there an efficient procedure to recognize whether a (0, 1) matrix is a member of this class?

To this end, define a (0, 1) matrix to be *totally balanced* if it does not contain a square submatrix with row and column sums equal to 2 and no identical columns. Totally balanced matrices belong to the larger class of balanced matrices introduced by Berge (1972).

It is a simple observation that each matrix in standard greedy form is totally balanced (see Exercise 6.10). We will now prove the converse statement. Namely, each totally balanced matrix can be permutated into standard greedy form. This will characterize totally balanced matrices as the class of matrices which are permutable into standard greedy form. (Note that the property of total-balancedness is not affected by row or column permutations). The proof uses the concept of a lexical matrix which was introduced by Hoffman, Kolen, and Sakarovitch (1985).

We call a (0, 1) matrix *lexical* if both the rows and columns are ordered in a lexical nondecreasing order. (See Section 6.2.2 for the definition of lexical nondecreasing order).

Let the matrices A_1, A_2 be given by

$$A_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

Then matrix A_1 is totally balanced but not lexical; on the other hand, the matrix A_2 is lexical but not totally balanced. Thus the class of lexical matrices does not coincide with the class of totally balanced matrices.

An algorithm that transforms any $m \times n$ (0, 1) matrix into a lexical matrix by permuting rows and by permuting columns of the matrix is presented by Hoffman et al. (1985). The algorithm consumes $O(nm^2)$ time (assuming that $m \leq n$).

In the theorem below we show that a totally balanced matrix which is lexical is indeed in standard greedy form. In view of the above algorithm and the fact that total-balancedness is not affected by row and column permutations, the theorem implies that a totally balanced matrix is transformable by row and column permutations into standard greedy form.

Theorem 6.10. *Let $A = (a_{ij})$ be a totally balanced lexical matrix. Then A is in standard greedy form.*

Proof. Suppose that A is not in standard greedy form. Then there exist rows $i(1)$, $i(2)$, (with $i(1) < i(2)$) and columns $j(1)$, $j(2)$ (with $j(1) < j(2)$) such that $a_{i(1), j(1)} = a_{i(1), j(2)} = a_{i(2), j(1)} = 1$ and $a_{i(2), j(2)} = 0$. Let $i(3)$ be the last row in which columns $j(1)$, $j(2)$ differ, and let $j(3)$ be the last column in which rows $i(1)$ and $i(2)$ differ. Since A is lexical, $i(3) > i(2)$, $j(3) > j(2)$ and $a_{i(3), j(1)} = 0$, $a_{i(3), j(2)} = 1$, $a_{i(1), j(3)} = 0$, $a_{i(2), j(3)} = 1$. Furthermore since A is totally balanced, it does not contain a 3×3 submatrix with row and column sums equal to two. We can therefore conclude that $a_{i(3), j(3)} = 0$.

Consider now a square submatrix B of A of maximum order, say k, satisfying the following property: if $i(1) < i(2) < \cdots < i(k)$ and $j(1) < j(2) < \cdots < j(k)$ denote, respectively, the indices of the rows and columns of B, then

$$
\begin{array}{c} \\ i(1) \\ i(2) \\ i(3) \\ \vdots \\ i(k) \end{array}
\begin{array}{c}
\begin{array}{cccccc} j(1) & j(2) & j(3) & \cdots & & j(k) \end{array} \\
\begin{bmatrix}
1 & 1 & 0 & \cdots & & 0 \\
1 & 0 & 1 & & & \\
0 & 1 & 0 & & & \\
\vdots & & & & & 0 \\
\vdots & & & & & 1 \\
0 & & & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

1. $i(p)$, $p = 3, \ldots, k$, is the last row in which columns $j(p-2)$ and $j(p-1)$ differ,
2. $j(p)$, $p = 3, \ldots, k$, is the last column in which rows $i(p-2)$ and $i(p-1)$ differ,
3. B has ones only in the lower and upper diagonal and the first element of the main diagonal and zero elsewhere.

(Note that $k \geq 3$ by the above argument.)

Using the fact that A is lexical define $i(k+1)$ $(>i(k))$ to be the last row in which columns $j(k-1)$ and $j(k)$ differ, and let $j(k+1)$ $(>j(k))$ be the last column in which rows $i(k-1)$ and $i(k)$ differ. We must have $a_{i(k+1)j(k-1)} = 0$, $a_{i(k+1)j(k)} = 1$, $a_{i(k-1)j(k+1)} = 0$, $a_{i(k)j(k+1)} = 1$. By the definition of $i(p)$ and $j(p)$ $(3 \leq p \leq k)$, we know that $a_{i(k+1)j(q)} = 0$ and $a_{i(q)j(k+1)} = 0$, $q = 1, 2, \ldots, k-1$. Again, using the fact that A is totally balanced we know that A does not contain a $(k+1) \times (k+1)$ submatrix (with nonidentical columns) whose row and column sums are equal to two. Thus $a_{i(k+1)j(k+1)} = 0$, and the submatrix defined by the rows $i(1) < i(2) < \cdots < i(k+1)$ and columns $j(1) < j(2) < \cdots < j(k+1)$ contradicts the maximality of B. □

As a consequence of the above theorem we conclude that an $m \times n$ matrix A is totally balanced if and only if the $O(nm^2)$ algorithm, which transforms any $m \times n$ $(0, 1)$ matrix into a lexical matrix, transforms A into standard greedy form. Testing whether a matrix is in standard greedy form can also be performed in $O(nm^2)$ time comparing each pair of rows. Testing for total-balancedness can therefore be done in $O(nm^2)$ time. Algorithms with lower complexity bounds have been developed recently by Lubiw (1987) and Paige and Tarjan (1987).

In the case of the covering problem on trees, the standard greedy form was obtained in $O(mn)$ and $O(mn + m \log m)$ times for the client constrained covering problem and the facility constrained covering problems,

respectively (see Section 6.2.2). In particular the matrix A associated with this problem on a tree is also totally balanced. This matrix A is the intersection matrix of neighborhood subtrees versus nodes. The fact that such a matrix is totally balanced was first proved by Giles (1978). This result was generalized by Tamir (1983) who proved that the intersection matrix of neighborhood subtrees versus neighborhood subtrees is totally balanced. This latter result can also be proved using the lexical representation of a totally balanced matrix (see Hoffman et al., 1985). However, the converse that "every totally balanced matrix is the intersection matrix of neighborhood subtrees versus neighborhood subtrees of some tree," was proved to be false by Broin and Lowe (1986) who gave the following counterexample of such a matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Recently, Tamir (1987) has presented other classes of totally balanced matrices defined by center location problems on trees.

In the remainder of this section we will discuss briefly some covering problems defined by totally balanced matrices.

The budget constrained *coverage problem* is to minimize the penalty costs given an upper bound on the setup costs. In case of equal setup costs the budget constraint reduces to an upperbound on the number of facilities say p. This problem can be formulated as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} p_i z_i \\ \text{subject to} \quad & \sum_{j=1}^{m} a_{ij} x_j + z_i \geq 1, \qquad i = 1, \ldots, m, \\ & \sum_{j=1}^{m} x_j \leq p \qquad (6.4.1) \\ & x_j \in \{0, 1\}, \qquad j = 1, \ldots, m, \\ & z_i \in \{0, 1\}, \qquad i = 1, \ldots, m \end{aligned}$$

where $a_{ij} = 1$ if client i can be served by a facility at v_j, $a_{ij} = 0$ otherwise.

The p-median problem on a tree turns out to be a special case of (6.4.1). The p-median problem is to locate p points $X = \{x_1, \ldots, x_p\}$ on the tree in order to minimize the sum of the transportation costs. In the case of linear transportation costs this problem can be formulated as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} w_i D(v_i, X) \\ \text{subject to} \quad & X \subseteq T, \\ & |X| = p \end{aligned} \tag{6.4.2}$$

where w_i is a nonnegative weight corresponding to client i, $i = 1, \ldots, m$. It was proved by Hakimi (1965) that there exists an optimal solution with $X \subseteq V$. With this in mind we can easily verify that the p-median problem can be reformulated very similar to the UFL problem in Section 6.2.1 as a client constrained coverage problem. The reader is asked to construct such a formulation (see Exercise 6.13). Kariv and Hakimi (1979b) gave an $O(m^2p^2)$ dynamic programming approach to solve the p-median problem on trees.

A second example is the maximum coverage problem considered by Megiddo et al. (1983). In this scenario they want to locate new facilities on the tree. In case a new facility is within distance r_i of client i this client will go to that new facility and this will generate a revenue of p_i, $i = 1, \ldots, m$. The *maximum coverage problem* is to establish at most p facilities on the tree so as to maximize the total revenue obtained from clients who will use the new facilities (equivalently, minimize the sum of revenues of clients who will not use the new facilities). It can be shown (as in the p-median problem), that facility locations may be restricted to a finite set of points on the tree. If the nodes are identified as clients, then the locations can be limited only to points on the tree that belong to the intersection of a maximal subset of neighborhood subtrees in the collection $N(v_i, r_i)$, $i = 1, \ldots, m$. It suffices to consider one point in the intersection of each maximal subset of neighborhood subtrees. From the discussion in Section 6.3, there is a one to one correspondence between maximal subsets of neighborhood subtrees (with nonempty intersection) and the maximal cliques of the intersection graph G of the subtrees $N(v_i, r_i)$, $i = 1, \ldots, m$. Therefore each maximal clique of G will contribute one potential facility location. Since G is a chordal graph with m nodes, it has at most m maximal cliques (see Exercise 6.8), and we can restrict our attention only to at most m potential locations corresponding to the maximal cliques.

Using the finiteness of the set of potential facility sites, the maximum coverage problem can be formulated as (6.4.1), and like the p-median problem it can be solved in $O(m^2p^2)$ time.

Therefore we have two instances of problem (6.4.1) where A is a totally balanced matrix and which can be solved in polynomial time. The question immediately arises whether we can solve (6.4.1) polynomially for any totally balanced matrix A?

If A is totally balanced, then the matrix given by

$$\begin{bmatrix} A & I \\ e & 0 \end{bmatrix}$$

where I is the identity matrix and e is a row vector of all ones, is also totally balanced. In contrast to the LP-relaxation of (6.2.2) the LP-relaxation of (6.4.1) does not always give an integer optimal solution. This is shown by the following example. Let A be the 7×7 matrix given by

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Take $p_i = 1$, $i = 1, \ldots, 6$, $p_7 = 0$, $p = 2$. An optimal solution of (6.4.1) is given by $x_1 = x_7 = 1$ and optimal value 2. If we take $x_1 = x_2 = x_3 = x_7 = \frac{1}{2}$ we get a solution of 3/2. This example arises with the tree shown in Figure 6.7.

Broin and Lowe (1986) give an $O(mn^3 + p^2n^3)$ dynamic programming algorithm to solve problem (6.4.1), when its matrix is of size $m \times n$ and $n \leq m$.

Consider the dominating set problem on a graph. Given a graph $G = (V, E)$ and an integer $k \leq |V|$ the *dominating set problem* is to determine whether there exists a subset of at most k nodes such that each node not in this subset is adjacent to at least one of these nodes. It is well known that this problem is $\mathcal{NP}$-complete even for chordal graphs (see Booth and Johnson, 1982).

Let us generalize the dominating set problem on a graph. Consider a graph $G = (V, E)$ with nodes v_i, $i = 1, \ldots, m$ and all arc lengths equal to one. With each node v_i we associate a nonnegative integer r_i and a cost c_i, $i = 1, \ldots, m$. The *generalized dominating set problem* is to minimize the total cost of a subset of nodes such that each node v_i, $i = 1, \ldots, m$ is within distance r_i from at least one of the nodes in this subset. This problem can be formulated as

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{m} c_j x_j \\ &\text{subject to} && \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1, \ldots, m, \\ & && x_j \in \{0, 1\}, \qquad j = 1, \ldots, m \end{aligned} \tag{6.4.3}$$

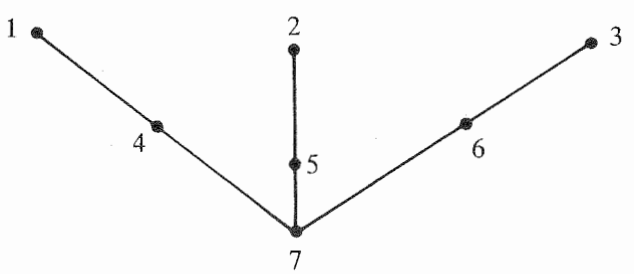

Figure 6.7. Arc lengths equal to one, $r_i = 1$, $i = 1, \ldots, 7$.

where $a_{ij} = 1$ if $d(v_i, v_j) \leq r_i$, $a_{ij} = 0$ otherwise. When $r_i = 1$ and $c_i = 1$, $i = 1, \ldots, m$ we get the dominating set problem.

There exists a subclass of chordal graphs for which the generalized dominating set problem can be solved in polynomial time. The *neighborhood matrix* of a graph $G = (V, E)$ with nodes $v_1, \ldots, v_m$ is the $m \times m$ $(0, 1)$ matrix defined by $a_{ii} = 1$, $i = 1, \ldots, m$ and $a_{ij} = 1$, $i \neq j$, if and only if $[v_i, v_j] \in E$. A graph is a *strongly chordal graph* if and only if its neighborhood matrix is totally balanced. For a given graph with nodes $v_1, \ldots, v_m$ and all arc lengths equal to 1, let us define the $m \times m$ $(0, 1)$ matrix A^k by $a_{ij}^k = 1$ if and only if $d(v_i, v_j) \leq k$. The neighborhood matrix is equal to A^1. It was proved by Lubiw (1982) that if A is totally balanced, then $[I|A^2|\cdots|A^k]$ ($[A|B]$ denotes the matrix where the first m columns are the columns of the matrix A and the last m columns are the columns of B) is totally balanced for every $k \geq 1$ (see Exercise 6.14). From this it follows that whenever we have a strongly chordal graph the matrix of (6.4.3) is totally balanced and hence we can solve (6.4.3) using the algorithm of Section 6.2.1 in $O(m^3)$ time by first transforming it into standard greedy form.

If we allow the arcs of the strongly chordal graph to have nonunit lengths, then problem (6.4.3) becomes $\mathcal{NP}$-hard even for the case where arcs have length equal to 1 or 2. The dominating set problem is reducible to this problem. Given a graph G with unit arc lengths, transform it into a complete graph (which is clearly strongly chordal) by augmenting all missing arcs and assigning each one of them a length of 2. The dominating set problem is then equivalent to the generalized dominating set problem (6.4.3) with $a_{ij} = 1$ if and only if $d(v_i, v_j) \leq 1$.

For further relationships between totally balanced matrices and strongly chordal graphs the reader is referred to Anstee and Farber (1984), Farber (1983, 1984), Chang (1982), Iijima and Shibata (1979), and Lubiw (1982).

ACKNOWLEDGMENT

This chapter was written with the support of the Erasmus University Rotterdam during the period Arie Tamir was a visiting professor at Erasmus University.

EXERCISES

6.1 Let $N(x, r)$ be a neighborhood subtree of a given tree T, and let t be a tip node of T. Define T_1 to be the subtree obtained from T by removing t and the unique arc adjacent to it. Prove that $N(x, r) \cap T_1$ is a neighborhood subtree of T_1.

6.2 Let v be a given node of a tree T. Prove that a node which is at a largest distance from v is an endpoint of a longest path in T.

6.3 Let t_1 and t_2 be the two endpoints of a longest (simple) path on a given tree T. Prove that the unique solution to the (unweighted) 1-center problem on T is obtained by setting the center at the middle point of the path $P[t_1, t_2]$.

6.4 Let t_1 and t_2 be the two endpoints of a longest (simple) path on a given tree T. Show that if a neighborhood subtree $N(x, r)$ contains t_1 and t_2, then $N(x, r) = T$.

Exercises 6.5, 6.6, and 6.7 deal with p-center problems on the real line. In all these problems assume that $v_1 < v_2 < \cdots < v_m$ are given real points on the line and $w_1, w_2, \ldots, w_m$ are nonnegative reals. The weighted p-center problem is given by

$$\begin{aligned} &\text{minimize} && z \\ &\text{subject to} && \min_{j=1,\ldots,p} \{w_i|v_i - x_j|\} \le z\,, \qquad i = 1, \ldots, m\,, \qquad (*) \\ & && x_1, x_2, \ldots, x_p \text{ reals} \end{aligned}$$

6.5 Formulate the weighted 1-center problem on the line as a two-dimensional linear program with at most $2m$ constraints.

6.6 Consider the unweighted version of (*), that is, $w_i = 1$, for $i = 1, \ldots, m$. Let z^* denote the optimal value of z in (*). Design an algorithm of order $O(p \log m)$ to test whether or not z^* is greater than a given value r (this is equivalent to testing whether p-centers will suffice to cover each v_i, $i = 1, \ldots, m$ within a radius r).

6.7 Given a positive number r, define the $m \times m$ incidence matrix $A = (a_{ij})$ as follows

$$a_{ij} = \begin{cases} 1\,, & \text{if } w_i|v_i - v_j| \le r\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

(a) Prove that A is totally balanced.

(b) Prove that every square nonsingular submatrix of A has a determinant ± 1, that is, A is totally unimodular.

6.8 Prove that a chordal graph on m nodes has at most m maximal cliques.

6.9 Construct an efficient algorithm that solves the budget constrained center problem on a cycle network G, that is, $G = (V, A)$ where

$$V = \{v_1, \ldots, v_m\}$$

and

$$A = \{[v_1, v_2], [v_2, v_3], \ldots, [v_{m-1}, v_m], [v_m, v_1]\}\,.$$

6.10 Show that a matrix in standard greedy form is totally balanced.

6.11 Let $V = \{v_1, \ldots, v_m\}$ be the node set of a tree T. Let $v_1, \ldots, v_k$ be the set of tip nodes. Let $P_i = P[v_1, v_i]$, $i = 1, \ldots, k$ denote the simple path connecting v_1 with v_i. Let $\{Q_1, \ldots, Q_n\}$ be a collection of paths on T such that each Q_j, $j = 1, \ldots, n$ is contained in some path of the collection $\{P_1, \ldots, P_k\}$. Define the $n \times m$ incidence matrix $A = (a_{ij})$ by

$$a_{ij} = \begin{cases} 1, & \text{if } v_j \in Q_i, \\ 0, & \text{otherwise.} \end{cases}$$

Prove that A is totally balanced. (Hint: Prove the nest ordering property and use Lemma 6.1 and Exercise 6.10.)

6.12 Let $T = (V, A)$, $V = \{v_1, \ldots, v_m\}$ be a tree. For each pair v_i and v_j of adjacent nodes on T replace the (undirected) arc $[v_i, v_j]$ by two directed arcs $[v_i, v_j]$ and $[v_j, v_i]$. Assign arbitrary nonnegative lengths to the directed arcs. The network resulting is called a *bitree*. For v_i, v_j let $d(v_i, v_j)$ denote the length of the shortest directed path from v_i to v_j. Given nonnegative reals $r_1, \ldots, r_m$, define the *outer neighborhood* $N(v_j, r_j) = \{v_i \in V: d(v_j, v_i) \leq r_j\}$, $j = 1, \ldots, m$. Prove that the $m \times m$ incidence matrix of outer neighborhood versus nodes is totally balanced.

6.13 Formulate the p-median problem as a client constrained coverage problem (see Section 6.4).

6.14 Let A be a neighborhood matrix of a graph. Suppose that A is totally balanced. Prove that the matrix $[I|A|A^2|\cdots|A^k]$ is totally balanced for all $k \geq 1$ (see Section 6.4 for the definition of A^i).

REFERENCES

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1974). *The Desing and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachusetts.

Anstee, R. P., and M. Farber (1984). "Characterizations of Totally Balanced Matrices." *Journal of Algorithms* **5**, 215–230.

Berge, C. (1972). "Balanced Matrices." *Mathematical Programming* **2**, 19–31.

Booth, K. S., and J. H. Johnson (1982). "Dominating Sets in Chordal Graphs." *SIAM Journal on Computing* **11**, 191–199.

Broin, M. W., and T. J. Lowe (1986). "A Dynamic Programming Algorithm for Covering Problems with (Greedy) Totally-Balanced Constraint Matrices." *SIAM Journal on Algebraic and Discrete Methods* **7**, 348–357.

Buneman, P. (1974). "A Characterization of Rigid Circuit Graphs." *Discrete Mathematics* **9**, 205–212.

Chandrasekaran, R., and A. Daughety (1981). "Location on Tree Networks, p-Center and N-Dispersion Problems." *Mathematics of Operations Research* **6**, 50–57.

Chandrasekaran, R., and A. Tamir (1980). "An $O((n \log p)^2)$ Algorithm for the Continuous p-Center Problem on a Tree." *SIAM Journal on Algebraic and Discrete Methods* **1**, 370–375.

Chandrasekaran, R., and A. Tamir (1982). "Polynomially Bounded Algorithms for Locating p-Centers on a Tree." *Mathematical Programming* **22**, 304–315.

Chang, G.J. (1982). *K-Domination and Graph Covering Problems*. Ph.D. Dissertation, Cornell University, Ithaca, New York.

Farber, M. (1983). "Characterization of Strongly Chordal Graphs." *Discrete Mathematics* **43**, 173–189.

Farber, M. (1984). "Domination, Independent Domination and Duality in Strongly Chordal Graphs." *Discrete Applied Mathematics* **7**, 115–130.

Frederickson, G. N., and D. B. Johnson (1983). "Finding k-th Paths and p-Centers by Generating and Searching Good Data Structures." *Journal of Algorithms* **4**, 61–80.

Gavril, F. (1972). "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph." *SIAM Journal on Computing* **1**, 180–187.

Gavril, F. (1974). "The Intersection Graphs of Subtrees are Exactly the Chordal Graphs." *Journal of Combinatorial Theory, Series B* **16**, 47–56.

Giles, R. (1978). "A Balanced Hypergraph Defined by Subtrees of a Tree." *ARS Combinatorica* **6**, 179–183.

Golumbic, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

Hajnal, A., and J. Súranyi (1958). "Uber die Auflösung von Graphen in vollständige Teilgraphen." *Annales Universitatis Scientiarum Budapestinensis Eötvös Nominate, de Rolando Sectio Mathematica* **1**, 113–121.

Hakimi. S. L. (1965). "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems." *Operations Research* **13**, 462–475.

Handler, G. Y. (1973). "Minimax Location of a Facility in an Undirected Tree Graph." *Transportation Science* **7**, 287–293.

Hoffman, A. J., A. Kolen, and M. Sakarovitch (1985). "Totally Balanced and Greedy Matrices." *SIAM Journal on Algebraic and Discrete Methods* **6**, 721–730.

Iijima, K., and Y. Shibata (1979). *A Bipartite Representation of a Triangulated Graph and Its Chordality*, *CS*-79-1. Department of Computer Science, Gunma University, Japan.

Kariv, O., and S. L. Hakimi (1979a). "An Algorithmic Approach to Network Location Problems. Part 1. The p-Centers." *SIAM Journal on Applied Mathematics* **37**, 513–538.

Kariv, O., and S. L. Hakimi (1979b). "An Algorithmic Approach to Network Location Problems. Part 2. The p-Medians." *SIAM Journal on Applied Mathematics* **37**, 539–560.

Kolen, A. W. J. (1983). "Solving Covering Problems and the Uncapacitated Plant Location Problem on Trees." *European Journal of Operational Research* **12**, 266–278.

Kolen, A. W. J. (1985). "The Round-Trip p-Center and Covering Problem on a Tree." *Transportation Science* **19**, 222–234.

Lubiw, A. (1982). "Γ-Free Matrices." Master's Thesis, Faculty of Mathematics, University of Waterloo, Ontario, Canada.

Lubiw, A. (1987). "Doubly Lexical Ordering of Matrices." *SIAM Journal on Computing* **16**, 854–879.

Megiddo, N., and A. Tamir (1983). "New Results on p-Center Problems." *SIAM Journal on Computing* **12**, 751–758.

Megiddo, N., A. Tamir, E. Zemel, and R. Chandrasekaran (1981). "An $O(n \log^2 n)$ Algorithm for the k-th Longest Path in a Tree with Applications to Location Problems." *SIAM Journal on Computing* **10**, 328–337.

Megiddo, N., E. Zemel, and S. L. Hakimi (1983). "The Maximum Coverage Location Problem." *SIAM Journal on Algebraic and Discrete Methods* **4**, 253–261.

Paige, R., and R. E. Tarjan (1987). "Three Partition Refinement Algorithms." *SIAM Journal on Computing* **16**, 973–989.

Rose, D. J., R. E. Tarjan, and G. S. Lueker (1976). "Algorithmic Aspects of Vertex Elimination in Graphs." *SIAM Journal on Computing* **5**, 266–281.

Shier, D. R. (1977). "A Minimax Theorem for p-Center Problems on a Tree." *Transportation Science* **11**, 243–252.

Slater, P. S. (1976). "R-Domination in Graphs." *Journal of the Association for Computing Machinery* **23**, 446–450.

Tamir, A. (1980). *On the Core of Cost Allocation Games Defined on Location Problems*, Tech. Rep. Tel Aviv University, Tel Aviv, Israel.

Tamir, A. (1983). "A Class of Balanced Matrices Arising from Location Problems." *SIAM Journal on Algebraic and Discrete Methods* **4**, 363–370.

Tamir, A. (1987). "Totally Balanced and Totally Unimodular Matrices defined by Center Location Problems." *Discrete Applied Mathematics* **16**, 245–263.

Tamir, A., and E. Zemel (1982). "Locating Centers on Tree with Discontinuous Supply and Demand Regions." *Mathematics of Operations Research* **7**, 183–197.

Tansel, B. C., R. L. Francis, T. J. Lowe, and M. L. Chen (1982). "Duality and Distance Constraints for the Nonlinear p-Center Problem and Covering Problem on a Tree Network." *Operations Research* **30**, 725–744.

Wagon, S. (1978). "Infinite Triangulated Graphs." *Discrete Mathematics* **22**, 183–189.

Walter, J. R. (1972). *Representations of Rigid Cycle Graphs*. Ph.D. Dissertation, Wayne State University, Detroit, Michigan.

Walter, J. R. (1978). "Representations of Chordal Graphs as Subtrees of a Tree." *Journal of Graph Theory* **2**, 265–267.