Facility Location Problems on Tree Graphs with Different Speeds for Customers and Servers: A Study on Covering Problems Defined by Families of Subtrees

Arie Tamir[1].

June 6, 2000

Abstract: In the classical p-center location problems, the objective is to locate p servers (centers) minimizing the maximum distance or travel time between customers and their respective servers. We consider a generalization where in case of a call by a customer, the server and the customer start moving towards each other, possibly at different speeds. The time elapses till the customer meets the server is the response time. The objective is to minimize the maximum response time over all customers. Concentrating on tree networks, we study the related covering problems defined by collections of neighborhood subtrees (disks). We present efficient algorithms to solve these covering problems and the above generalized p-center model.

1 Introduction

In the classical p-center problem on a network there are customers located at the nodes of the network, and the objective is to locate p servers (centers) in order to minimize the maximum travel distance between the customers and their respective nearest servers. This model is typically applicable to emergency facilities like fire department teams and ambulances. If the nature of the service is such that customers travel to the servers, and each customer may move at a different speed, an appropriate objective is to minimize the

[1]Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel. atamir@math.tau.ac.il

maximum travel time. Assuming constant speeds, the latter case reduces to the classical weighted p-center problem, where distances are weighted by the customer speeds. A symmetric model, where servers travel from their home locations (possibly at different speeds), to the customers might be appropriate for other practical situations. We have been motivated by an application which unifies both models. Suppose that in an emergency situation a customer places a call for help. The customer has a vehicle that can start traveling immediately, in any direction at a given speed w. There are several available servers (emergency crews) located on the network, and they may also have different speeds. The objective is to minimize the response time, i.e., find a server that will instantenously start moving towards the customer, minimizing the travel time elapses till the customer meets the server. The problem is to find the location of p servers, minimizing the maximum response time to the customers. We are unware of any previous studies of this problem, where both customers and servers may move simultaneously at different velocities. In this paper we discuss this generalized center location problem, and study some of its variations, depending on the nature of the set of points, where customers and servers are allowed to meet. We note that since the classical unweighted p-center problem is NP-hard, so is the generalized model. Here we focus on tree networks and study the properties of the mathematical model. In particular, we present polynomial algorithms.

2 The Generalized p-center Problem

Let $T = (V, E)$ be an undirected tree network with node set $V = \{v_1, ..., v_n\}$ and edge set $E = \{e_2, ..., e_n\}$. Each edge e_j, $j = 2, 3, ..., n$, has a positive length l_j, and is assumed to be rectifiable. In particular, an edge e_j is identified as an interval of length l_j so that we can refer to its interior points. We assume that T is embedded in the Euclidean plane. Let $A(T)$ denote the continuum set of points on the edges of T. We view $A(T)$ as a connected and closed set which is the union of $n-1$ intervals. (A pair of intervals may intersect only at a common node.) Let $P(v_i, v_j)$ denote the unique simple path in $A(T)$ connecting v_i and v_j. Suppose that the tree T is rooted at some distinguished node, say v_1. For each node v_j, $j = 2, 3, ..., n$, let $p(v_j)$, the *parent* of v_j, be the node $v \in V$, closest to v_j, $v \neq v_j$ on $P(v_1, v_j)$. v_j is a *child* of $p(v_j)$. e_j is the edge connecting v_j with its parent $p(v_j)$. A node v_i is a *descendant* of v_j if v_j is on $P(v_i, v_1)$. A node of T is a *leaf node* if it is incident to exactly one edge of T. Such an edge is called a *leaf edge*.

We refer to interior points on an edge by their distances along the edge from the two nodes of the edge. The edge lengths induce a distance function on $A(T)$. For any pair of points $x, y \in A(T)$, we let $d(x, y)$ denote the length of $P(x, y)$, the unique simple path in $A(T)$ connecting x and y. If x and y are on the same edge, $P(x, y)$ is called a *subedge* or a *partial edge*, and its length is $d(x, y)$. Generally, the path $P(x, y)$ is also viewed as a collection of edges and at most two subedges (partial edges). Also, for any pair of subsets $X, Y \subseteq A(T)$, and $x \in A(T)$ we define $d(x, Y) = d(Y, x) =$ Infimum $\{d(x, y) | y \in Y\}$, and $d(X, Y) = d(Y, X) =$ Infimum $\{d(x, y) | x \in X, y \in Y\}$. $A(T)$ is a metric space with respect to the above distance function.

We now define the generalized center location model on the given tree. Each node v_i is identified as a customer with a travel velocity w_i. Also, each node v_j is identified as a potential location point for a server with travel velocity w'_j. If there is a call for service from customer v_i, both the customer and its designated server, say v_j, will start moving towards each other until they meet. The response time is the time elapses till the customer and the server meet. In the p-center problem for this model, the objective is to minimize the maximum response time with p servers. We consider two versions. In the continuous version, the meeting point can be any point of $A(T)$, where in the discrete version a customer and a server can meet only at a node. To formulate a mathematical framework we will need more definitions.

A subset $Y \subseteq A(T)$ is called a *subtree* if it is closed and connected. Y is also viewed as a finite (connected) collection of partial edges (closed subintervals), such that the intersection of any pair of distinct partial edges is empty or is a point in V. We call a subtree *discrete* when all its (relative) boundary points are nodes of T. Note that a discrete subtree can also be viewed as a subgraph of T.

Given a nonnegative real r, and a node v_i, we let $T(v_i, r)$ denote the subtree consisting of all points in $A(T)$ whose distance from v_i is at most r, i.e., $T(v_i, r) = \{x \in A(T) | d(v_i, x) \leq r\}$. $T(v_i, r)$ is called the *neighborhood* or *disk* of radius r centered at v_i. Similarly, $Tr(v_i, r)$, the *truncated neighborhood* or *truncated disk* of radius r centered at v_i is defined as the subgraph of T induced by the node set $V(v_i, r) = \{v_j | d(v_i, v_j) \leq r\}$. $Tr(v_i, r)$ is a subset of $T(v_i, r)$, and it is also viewed as a discrete subtree of $A(T)$. We also call $Tr(v_i, r)$ a *discrete disk* of radius r centered at v_i.

Let t be a positive real, denoting the response time for the customers. Let $p(t)$ be the minimum number of servers (nodes) needed to ensure a maximum response time t for all customers (nodes). $p(t)$ is clearly a monotone nonincreasing function of t. t^*, the optimal value for the maximum response time that can be ensured with p servers is then the smallest value of t, such that $p(t) \leq p$. Formally, for the continuous version of the generalized model, $p(t)$ is defined by

$$p(t) = \{\min_{V' \subset V} |V'| : T(v_i, tw_i) \cap (\cup_{v_j \in V'} \{T(v_j, tw'_j)\}) \neq \emptyset, i = 1, ..., n\}.$$

Similarly, for the discrete version of the generalized model, $p(t)$ is defined by

$$p(t) = \{\min_{V' \subset V} |V'| : Tr(v_i, tw_i) \cap (\cup_{v_j \in V'} \{Tr(v_j, tw'_j)\}) \neq \emptyset, i = 1, ..., n\}.$$

For the continuous model, if v_i is served by the server located at v_j, the response time for v_i is $d(v_i, v_j)/(w_i + w'_j)$. Therefore, the value of t^* for the continuous generalized p-center problem is given by

$$t^* = \min_{V' \subset V : |V'| = p} \max_{v_i \in V} \{ \min_{v_j \in V'} d(v_i, v_j)/(w_i + w'_j) \}.$$

We conclude that t^* for the continuous model is an element in the set $R_1 = \{d(v_i, v_j)/(w_i + w'_j) | i, j = 1, ..., n\}$. Similarly the value of t^* for the discrete generalized p-center problem is clearly an element in the set $R_2 = \{d(v_i, v_k)/w_i | i, k = 1, ..., n\} \cup \{d(v_k, v_j)/w'_j | k, j = 1, ..., n\}$. The structure of the set R_2 enables us to use the search procedure in Megiddo et al. (1981) to find t^* in the discrete case by computing $p(t)$ for $O(\log n)$ values of the parameter t. Similarly, using the structure of the set R_1, we can apply the search in Megiddo and Tamir (1983) to find t^* in the continuous case by computing $p(t)$ for $O(\log^2 n)$ values of the parameter t. Note that $p(t)$ is the solution value of a covering problem of disks by disks. In the following sections we show how to solve these covering problems, as well as their weighted versions, efficiently for both the continuous and the discrete models. In the last section we discuss some related models and generalizations.

3 The Covering Models

Suppose that each node $v_i \in V$ is associated with two nonnegative reals r_i and s_i. Consider the four collections of disks, $S_1^* = \{T(v_j, r_j) | j = 1, ..., n\}$, $S_2^* = \{Tr(v_j, r_j) | j = 1, ..., n\}$, $D_1^* = \{T(v_i, s_i) | i = 1, ..., n\}$, and $D_2^* = \{Tr(v_i, s_i) | i = 1, ..., n\}$.

In the next two sections we study weighted covering problems of one collection of disks (truncated disks) by the other collection of disks (truncated disks). Specifically, we assume that each node $v_j \in V$ has a nonnegative weight, a_j. In the *continuous disk covering problem* the objective is to find a subset of nodes (centers) V' of minimum total weight, such that the union of the subcollection of disks $\{T(v_j, r_j)|v_j \in V'\}$ intersects each disk in D_1^*. Similarly, in the *discrete disk covering problem* the objective is to find a subset of nodes (centers) V' of minimum total weight, such that the union of the subcollection of discrete disks $\{Tr(v_j, r_j)|v_j \in V'\}$ intersects each discrete disk in D_2^*. Note that the problem of computing $p(t)$, mentioned above corresponds to the special case where $s_i = tw_i$, $i = 1, ..., n$, $r_j = tw_j$, $j = 1, ..., n$, and $a_j = 1$ for $j = 1, ..., n$. For a pair v_j, v_i, we will say that the disk $T(v_j, r_j)$ $(Tr(v_j, r_j))$ *covers* or *pierces* the disk $T(v_i, s_i)$ $(Tr(v_i, s_i))$ if the respective pair of disks intersect. Note that the discrete disk $Tr(v_j, r_j)$ intersects the discrete disk $Tr(v_i, s_i)$ if and only if the continuous disk $T(v_j, r_j)$ intersects the discrete disk $Tr(v_i, s_i)$. Therefore, the discrete covering problem is equivalent to the problem of finding a subset of nodes (centers) V' of minimum total weight, such that the union of the subcollection of disks $\{T(v_j, r_j)|v_j \in V'\}$ intersects each discrete disk in D_2^*.

Several special cases of the above covering problems have been considered in the literature in the context of the p-center problem on a tree. Kariv and Hakimi (1979) considered the case where $r_j = 0$, $j = 1, ..., n$, i.e., each disk $T(v_j, r_j)$ shrinks to a single point. The problem reduces to covering disks with points (nodes). For the unweighted case, $(a_j = 1,\ j = 1, ..., n)$, they presented a simple $O(n)$ algorithm. Their algorithm is based on a "bottom-up" dynamic programming approach, starting with the leaves of the tree, and terminating at its root. The weighted version of this case was solved in $O(n^2)$ time in Kolen (1983), (see also Kolen and Tamir (1990)). We will show a simple transformation of the general continuous covering problem to the above special case where $r_j = 0$, $j = 1, ..., n$. We will then consider the discrete case, and present an $O(n^2)$ algorithm for its solution.

3.1 Covering (Continuous) Disks with (Continuous) Disks

The following reduction theorem indicates that the problem of covering disks by disks can be reduced to a problem of covering disks by points (points by disks). In particular, the problem of covering points by disks can be transformed into an equivalent problem of covering disks by points. Also, every incidence matrix of two families of disks on a tree, is also the incidence matrix of a family of disks and a subset of nodes of some tree.

Theorem 3.1 *Given the tree $T = (V, E)$, $V = \{v_1, ..., v_n\}$, and the two families of disks $\{T(v_j, r_j) | j = 1, ..., n\}$ and $\{T(v_i, s_i) | i = 1, ..., n\}$, for each $i, j = 1, ..., n$, define $a_{i,j}$ to be 1 if the pair of disks $T(v_i, s_i)$ and $T(v_j, r_j)$ intersect, and 0 otherwise. Let $A = (a_{i,j})$, be the respective incidence matrix. There exists a tree $T' = (V \cup U, E')$, $U = \{u_1, ..., u_n\}$, and a family of disks $\{T'(v_j, r'_j) | j = 1, ..., n\}$, in T', such that for each $i, j = 1, ..., n$, $a_{i,j} = 1$ if and only if u_i is in $T'(v_j, r'_j)$. Also, there exists a tree $T" = (V \cup U, E')$, $U = \{u_1, ..., u_n\}$, and a family of disks $\{T'(v_i, s'_i) | i = 1, ..., n\}$, in $T"$, such that for each $i, j = 1, ..., n$, $a_{i,j} = 1$ if and only if u_j is in $T'(v_i, s'_i)$.*

Proof:

We start by defining the tree T'. Define $r = 1 + \max\{r_j | j = 1, ..., n\}$ and $s = 1 + \max\{s_i | i = 1, ..., n\}$. T' is obtained from T by augmenting the set U to the node set of T, and connecting the pair (v_i, u_i), $i = 1, ..., n$, by an edge of length $s - s_i$. For each $j = 1, ..., n$, the disk $T'(v_j, r'_j)$ is defined by its center v_j, and its radius $r'_j = r_j + s$.

Consider a node u_i, and a disk $T'(v_j, r'_j)$. Then, u_i is contained in this disk if and only if $d'(u_i, v_j)$, the distance in T' between u_i and v_j is at most r'_j, i.e., $d'(u_i, v_j) = d(v_i, v_j) + s - s_i \leq r'_j = s + r_j$. The latter inequality is equivalent to $d(v_i, v_j) \leq s_i + r_j$, which in turn holds if and only if the pair of disks $T(v_i, s_i)$ and $T(v_j, r_j)$ intersect in T.

The tree T" is defined in a similar way. The only difference between T' and T" is that in the latter the length of the edge connecting the pair (u_j, v_j), $j = 1, ..., n$, is $r - r_j$. For each $i = 1, ..., n$, the disk $T'(v_i, s'_i)$ is defined by its center v_i, and its radius $s'_i = s_i + r$. As above, it is easy to see that $T'(v_i, s'_i)$ contains u_j if and only if the pair of disks $T(v_i, s_i)$ and $T(v_j, r_j)$ intersect in T.

∎

As a result of the above theorem, we conclude that the weighted covering problem of continuous disks by continuous disks can be solved, after the transformation, in $O(n^2)$ time by the algorithm in Kolen (1983). The unweighted case is solvable in $O(n)$ by the algorithm in Kariv and Hakimi (1979). In particular we have an $O(n \log^2 n)$ algorithm to solve the continuous generalized p-center problem on trees by the algorithm in Megiddo and Tamir (1983).

4 Covering Discrete Disks with Discrete Disks

To solve the discrete covering problem we will use the results and machinery developed in Kolen (1983), Tamir (1983), Hoffman et al., (1985), and Lubiw (1987) on totally balanced and greedy matrices.

4.1 Totally Balanced and Greedy Matrices

Let $A = (a_{i,j})$ be an $m \times n$, $\{0, 1\}$ matrix. A is *balanced* if it does not have a square submatrix of odd size with exactly two nonzero entries in each column and each row. A is *totally balanced* if it does not have a square submatrix with nonidentical columns, and exactly two nonzero entries in each column and each row. A is *greedy* if there are no row indices $i_1 < i_2$ and column indices $j_1 < j_2$, such that $a_{i_1,j_1} = a_{i_1,j_2} = a_{i_2,j_1} = 1$ and $a_{i_2,j_2} = 0$. Totally balanced and greedy matrices have been studied extensively in the literature. It was shown in Hoffman et al., (1985), and Lubiw (1987), that A is totally balanced if and only if there are row and column permutations transforming A into a greedy form. Efficient algorithms to test whether a matrix is totally balanced, and convert such a matrix to greedy form appeared in the above references. The most efficient scheme for permuting a totally balanced matrix to greedy form is the $O(mn\log(mn))$ procedure of Paige and Tarjan (1987).

Weighted covering problems, defined by totally balanced matrices can be solved efficiently, due to the integrality of the extreme points of the feasible set. Moreover, if A is already in greedy form, this weighted covering problem,

$$\min\{\sum_{j=1}^{n} a_j x_j | \sum_{j=1}^{n} a_{i,j} x_j \geq 1, i = 1, ..., m; x \geq 0\},$$

can be solved greedily in $O(mn)$ time, Hoffman et al., (1985). Summarizing, we conclude that any weighted covering problem defined by a totally balanced matrix can be solved in $O(mn\log(mn))$ time.

Motivated by the above location problems, we focus here on totally balanced matrices defined by collections of subtrees. For these classes we obtain the greedy form in quadratic time, improving the above bound by a logarithmic factor. We avoid the use of the $O(mn\log(mn))$ algorithm of Paige and Tarjan, and solve the weighted covering problem in $O(mn)$ time.

Given the two collections of discrete disks, D_2^* and S_2^*, we define their $\{0,1\}$ incidence matrix $A = (a_{i,j})$ by the intersection relationship, i.e., $a_{i,j} = 1$, if and only if $Tr(v_i, s_i)$ intersects $Tr(v_j, r_j)$. The incidence matrix of two collections of (discrete) disks is known to be totally balanced, Tamir (1983) and Lubiw (1987). We will show that such a matrix can be converted to greedy form in $O(n^2)$ time. Combined with the algorithm of Hoffman et al., this will imply an $O(n^2)$ algorithm to solve the weighted covering problem of discrete disks. This in turn, will lead to an $O(n^2 \log n)$ algorithm for solving the discrete generalized p-center problem on trees, by applying a binary search on the set R_2 defined above. The conversion to greedy form is based on the following property which characterizes totally balanced matrices.

Nestedness Property

Let $A = (a_{i,j})$ be an $m \times n$, $\{0,1\}$ matrix. Let $A_{.j}$, $j = 1,...,n$, denote the j-th column of A. A has the *nestedness property* if there exists a row, say i', such that for each pair of columns $A_{.j}$ and $A_{.k}$, with $a_{i',j} = a_{i',k} = 1$, either $A_{.j} \leq A_{.k}$, or $A_{.j} \geq A_{.k}$. i' is then called a *simplicial row* of A.

A totally balanced matrix is characterized by the property that each one of its submatrices has the nestedness property. A totally balanced matrix A has the *nest ordering property* with respect to its rows, if its first row, $i = 1$, is a simplicial row of A, and for each $i = 2,...,m$, row i of A is a simplicial row of the submatrix A^i, obtained from A by deleting rows $1, 2, ..., i-1$ of A. Efficient methods to identify a simplicial row of a totally balanced matrix may lead to quadratic or even subquadratic algorithms to permute the rows of the matrix to form a nest ordering. It is shown in Kolen and Tamir (1990) that if we are already given a nest ordering, the transformation to greedy form can be achieved in $O(mn)$ time by using radix sort procedures. One example is the incidence matrix of nodes (rows) vs. disks (columns) in a tree, Kolen (1983).

Theorem 4.1 *Let A be the incidence matrix of nodes (rows) vs. disks (columns) in a tree. Let v_p and v_q satisfy*

$$d(v_p, v_q) = \max\{d(v_k, v_t) | k, t = 1, ..., n\}.$$

Then the two rows of A corresponding to the (leaves) v_p and v_q are simplicial rows.

In this case the nest ordering of the nodes (rows) can be obtained in $O(n \log n)$ time, Kolen and Tamir (1990). (Such an ordering is induced by the ordering of the distances of the nodes from any specified node.)

A second example is the incidence matrix of (continuous) disks vs. (continuous) disks. From the proof of Theorem 3.1, and the preceding result we obtain the following theorem.

Theorem 4.2 *Given the tree $T = (V, E)$, $V = \{v_1, ..., v_n\}$, and the two families of disks $\{T(v_j, r_j) | j = 1, ..., n\}$ and $\{T(v_i, s_i) | i = 1, ..., n\}$, for each $i, j = 1, ..., n$, define $a_{i,j}$ to be 1 if the pair of disks $T(v_i, s_i)$ and $T(v_j, r_j)$ intersect, and 0 otherwise. Let $A = (a_{i,j})$, be the respective incidence matrix. Let v_p and v_q satisfy*

$$d(v_p, v_q) - (s_p + s_q) = \max\{d(v_k, v_t) - (s_k + s_t) | k, t = 1, ..., n\}.$$

Then the two rows corresponding to the disks $T(v_p, s_p)$ and $T(v_q, s_q)$ are simplicial rows.

A nest ordering in this case can also be derived in $O(n \log n)$ time as in the previous case, due to the transformation in Theorem 3.1.

In the next section we prove a similar result for collections of discrete disks, and in the following section we identify simplicial rows for Trubin matrices.

4.2 Nestedness Property for Collections of Discrete Disks

For $i = 1, ..., n$, let S_i denote the discrete disk $Tr(v_i, s_i)$. For $j = 1, ..., n$, let T_j denote the discrete disk $Tr(v_j, r_j)$. Let $A = (a_{i,j})$ be the $n \times n$, $\{0, 1\}$ matrix, corresponding to the collections of discrete disks S_i, $i = 1, ..., n$, (rows), and T_j, $j = 1, ..., n$, (columns).

Theorem 4.3 *Consider the collection of discrete disks $\{S_i\}$, $i = 1, ..., n$, where S_i is centered at v_i and its radius is s_i. Define*

$$DT' = \max\{d(S_i, S_j) | i, j = 1, ..., n\}.$$

Let (p, q), satisfy $d(S_p, S_q) = DT'$, and

$$d(v_p, v_q) - (s_p + s_q) = \max\{d(v_t, v_k) - (s_t + s_k) | t, k = 1, ..., n; d(S_t, S_k) = DT'\}.$$

Suppose that $DT' > 0$. Then, the two rows of A corresponding to the discrete disks S_p and S_q satisfy the nestedness property.

Proof:

Since DT' is positive, the distance between S_p and S_q is positive, and it is equal to $d(v', v")$, where v' is a leaf node of S_p, and $v"$ is a leaf node of S_q. Consider a pair of disks T_j and T_k, and suppose that both intersect S_p.

Case I

Assume first that T_j intersects S_q as well. We claim that in this case T_j intersects each disk S_i, $i = 1, ..., n$, and therefore the nestedness property is satisfied. Equivalently, we prove that $d(v_j, S_i) \leq r_j$, for $i = 1, ..., n$.

To prove this claim we first observe that T_j must contain, $P(v', v")$, the path connecting v' and $v"$. Consider an arbitrary disk S_i. If S_i intersects $P(v', v")$, then it also intersects T_j. Hence, suppose that S_i does not intersect $P(v', v")$, and let u be the closest point to S_i on $P(v', v")$. From the maximality of the pair $v', v"$ it follows that $d(u, S_i) \leq \min\{d(u, S_p), d(u, S_q)\}$. Let $w(j)$ be the closest point to v_j, the center of T_j in $P(v', v")$. Assume without loss of generality that $w(j)$ is also on $P(v', u)$. Therefore,

$$d(v_j, S_i) \leq d(v_j, u) + d(u, S_i) \leq d(v_j, u) + d(u, S_q) = d(v_j, v") \leq r_j.$$

We conclude that in this case T_j intersects S_i, for each $i = 1, ..., n$.

Case II

To complete the proof it is sufficient to consider the case where neither T_j nor T_k intersect S_q. Let $u(j)(u(k))$, be the closest point in $T_j(T_k)$ to $v"$. Note that $u(j)$ and $u(k)$ are not necessarily nodes of the tree. Assume without loss of generality that $d(u(j), v") \leq d(u(k), v")$.

Case II(a): $u(j)$ and $u(k)$ are on $P(v', v")$.

We show that if T_k intersects a discrete disk S_i, then so does T_j. Let u be the closest point to S_i on $P(v', v")$. If u is in S_i, then clearly u is on $P(v', u(k))$, which belongs to T_j. Thus, suppose that u is not in S_i. From the maximality of the pair of disks S_p and S_q, it follows that $d(u, S_i) \leq d(u, S_p) = d(u, v')$.

Let $w(j)(w(k))$, be the closest point to $v_j(v_k)$ on $P(v', v")$.

Suppose first, that $w(j)$ is on $P(u, v")$. Then,

$$d(v_j, S_i) \leq d(v_j, u) + d(u, S_i) \leq d(v_j, u) + d(u, S_p) = d(v_j, S_p) = d(v_j, v') \leq r_j.$$

We conclude that T_j intersects S_i.

Assume that u is on $P(w(j), v")$.

If $u = w(k)$, then $d(u, S_i) \leq d(u, S_p) \leq d(u, u(k)) \leq d(u, u(j))$. Therefore

$$d(v_j, S_i) \leq d(v_j, u) + d(u, S_i) \leq d(v_j, u) + d(u, u(j)) = d(v_j, u(j)) \leq r_j,$$

and T_j intersects S_i.

If $u \neq w(k)$, consider two subcases:

When $w(j)$ is on $P(w(k), v")$ we have

$$d(v_k, w(j)) + d(w(j), S_i) = d(v_k, S_i) \leq d(v_k, u(k)) \leq d(v_k, w(j)) + d(w(j), u(k)).$$

Therefore, $d(w(j), S_i) \leq d(w(j), u(k))$. We conclude that

$$\begin{aligned} d(v_j, S_i) &\leq d(v_j, w(j)) + d(w(j), S_i) \leq d(v_j, w(j)) + d(w(j), u(k)) \\ &= d(v_j, u(k)) \leq d(v_j, u(j)) \leq r_j, \end{aligned}$$

and T_j intersects S_i.

When $w(k)$ is on $P(w(j), v")$, (and $u \neq w(k)$), we have

$$\begin{aligned} d(v_j, S_i) &\leq d(v_j, w(k)) + d(w(k), S_i) \leq d(v_j, w(k)) + d(w(k), u(k)) \\ &= d(v_j, u(k)) \leq d(v_j, u(j)) \leq r_j, \end{aligned}$$

and T_j intersects S_i.

Case II(b): $u(j)$ is on $P(v', v")$, but $u(k)$ is not on $P(v', v")$.

Suppose that T_k intersects S_i. Since $u(k)$ is not on $P(v', v")$, it follows that u, the closest point to S_i on $P(v', v")$ is $u = v'$. Moreover, v' must be in S_i, since otherwise, the distance between S_i and S_q would exceed $d(v', v") = DT'$. The path $P(v', u(j))$ is in T_j, and therefore T_j intersects S_i.

Case II(c): $u(j)$ and $u(k)$ are not on $P(v', v")$.

As in the previous case, we must have that v' is in S_i. Since $u(k)$ and $u(j)$ are not on $P(v', v")$, it follows that $u(k)(u(j))$ is the closest point to v' in $T_k(T_j)$. Define $u'(k)(u'(j))$ to be the closest node to v' in $T_k(T_j)$. Since we do not assume here that $d(u(j), v") \leq d(u(k), v")$, we may suppose without loss of generality that $d(u'(j), v') \leq d(u'(k), v')$.

Suppose that T_k intersects some discrete disk S_i. We will prove that T_j intersects S_i. As in the previous case, we must have that v' is in S_i. In particular, since v' is in S_i, we conclude that $u'(k)$ is also in S_i. If $u'(j)$ is on $P(u'(k), v')$, then $u'(j)$ is in S_i, and T_j intersects S_i. Suppose that $u'(j)$ is not on $P(u'(k), v')$, and let u' be the closest point to $u'(j)$ on $P(u'(k), v')$. $d(u'(j), v") \leq d(u'(k), v")$ implies $d(u'(j), u') \leq d(u'(k), u')$.

Let T' be the minimal subtree containing $\{v', u'(k), u'(j)\}$, and let $w(i)$ be the closest point in T' to v_i the center of S_i. Since v' and $u'(k)$ are in S_i, and $d(u'(j), u') \leq d(u'(k), u')$, we conclude that if $w(i)$ is not on $P(u'(k), u')$, $u'(j)$ is in S_i, and therefore T_j intersects S_i.

Suppose that $w(i)$ is on $P(u'(k), u')$ and $u'(j)$ is not in S_i. Then, since v' is in S_i, we must have $d(v', u') < d(u'(j), u')$.

From the above we have

$$d(v', u') < d(u'(j), u') \leq d(u'(k), u').$$

Let $w(p)$ be the closest point in T' to v_p, the center of S_p. From the fact that $u'(j)$ and $u'(k)$ are not on $P(v', v")$, it follows that both nodes are in S_p. Thus, T' is in S_p. It is easy to check that any location of $w(p)$ in T' would imply that

$$s_p - d(v_p, v') > s_i - d(v_i, v').$$

In particular, it follows that v' is also the closest node of S_i to $v"$, and therefore $d(S_i, S_q) = d(v', v") = DT'$. However, from the above, we contradict the maximality of the pair p, q,

$$\begin{aligned} d(v_i, v_q) - (s_i + s_q) &= d(v_i, v') + d(v', v_q) - (s_i + s_q) \\ &> d(v_p, v') + d(v', v_q) - (s_p + s_q) \\ &= d(v_p, v_q) - (s_p + s_q). \end{aligned}$$

■

Remark 4.4 *From the above proof it follows that the result of the above theorem holds for any pair (p, q), satisfying $d(S_p, S_q) = DT'$, and*

$$\begin{aligned} d(v_p, v_q) - (s_p + s_q) &= \max\{d(v_t, v_q) - (s_t + s_q) | t = 1, ..., n; d(S_t, S_q) = DT', S_t \cap S_p \neq \emptyset\}, \\ d(v_p, v_q) - (s_p + s_q) &= \max\{d(v_p, v_t) - (s_p + s_t) | t = 1, ..., n; d(S_p, S_t) = DT', S_q \cap S_t \neq \emptyset\}. \end{aligned}$$

The next example illustrates that for discrete disks it is not sufficient to select an arbitrary pair maximizing the distance between all pairs of discrete disks to ensure the nestedness property.

Example 4.5 *Consider the tree with four nodes, and $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4)\}$. The length of (v_1, v_4) is 2, while the other two edges have length 1. For $i = 1, ..., 4$, S_i is centered at v_i. The radius of S_4 is 0, and the radii of the other discrete disks are 1. The maximum distance between pairs of discrete disks is 2, and $d(S_4, S_1) = d(S_4, S_2) = d(S_4, S_3) = 2$. Nevertheless, the discrete disk S_1 does not satisfy the nestedness property. (Consider the two disks $T_2 = \{v_2\}$ and $T_3 = \{v_3\}$.)*

Theorem 4.6 *Consider the collection of discrete disks $\{S_i\}$, $i = 1, ..., n$, where S_i is centered at v_i and its radius is s_i. Define*

$$DT' = \max\{d(S_i, S_j) | i, j = 1, ..., n\}.$$

Suppose that $DT' = 0$. Then there exists a node u, contained in each disk S_i, $i = 1, ..., n$. Let S_p be a disk such that

$$s_p - d(v_p, u) = \min\{s_i - d(v_i, u) | i = 1, ..., n\}.$$

Then, S_p satisfies the nestedness property.

Proof:

When $DT' = 0$, each pair of disks have a common point. From the Helly property on trees (see Kolen and Tamir (1990)), all disks share a common node, say u. Let T'' be the tree obtained from the given tree T, by augmenting a node, say v_{n+1}, and connecting it to u with an edge of length $s = 1 + \max\{s_i | i = 1, ..., n\}$. Define a new disk S_{n+1}, by its center, v_{n+1}, and $s_{n+1} = 0$. Applying the previous theorem to T'', and the respective collection of disks, we conclude that the pair of disks, S_{n+1} and S_p, defined above have the nestedness property with respect to T''. In particular, S_p satisfies the property with respect to T.

■

4.3 Converting an Incidence Matrix of Collections of Discrete Disks to Greedy Form

In this section we use the above results to obtain an $O(n^2)$ algorithm for transforming an incidence matrix of two collections of discrete disks into greedy form. As mentioned earlier, it is sufficient to find in $O(n^2)$ time a permutation of the rows of this incidence matrix that yields a nest ordering.

Using the notation in Theorem 4.3, for each $i, j = 1, ..., n$, define $d_{i,j} = d(S_i, S_j)$. Let D be the square matrix with elements $(d_{i,j})$, $i, j = 1, ..., n$. For each $j = 1, ..., n$, let $L(j)$ be the list obtained from the set $\{d_{i,j}\}$, $i = 1, ..., n$, by sorting the elements from largest to smallest, with the provision, that if $d_{i_1,j} = d_{i_2,j}$, and $d(v_{i_1}, v_j) - (s_{i_1} + s_j) < d(v_{i_2}, v_j) - (s_{i_2} + s_j)$, then the element $d_{i_2,j}$ precedes the element $d_{i_1,j}$.

The incidence matrix of the two collections of discrete disks can clearly be computed in $O(n^2)$ time from the lists $\{L(j)\}$, $j = 1, ..., n$. Moreover, Theorem 4.3 implies that the nest ordering of the rows can easily be derived from these lists in $O(n^2)$ time. For example, to obtain the first row in the nest ordering, we look at the (multi) set K consisting of the following n elements; the first element of $L(1)$, the first element of $L(2)$, etc. Let $K' = \{d_{i_1,j_1}, ..., d_{i_m,j_m}\}$ be the (multi) subset of K consisting of all the largest elements in K, where

$$d(v_{i_1}, v_{j_1}) - (s_{i_1} + s_{j_1}) = \max\{d(v_{i_t}, v_{j_t}) - (s_{i_t} + s_{j_t}) | t = 1, ..., m\}.$$

Then d_{i_1,j_1}, defines the discrete disk S_{j_1} as the one corresponding to the first row of the nest ordering. To obtain the second row, omit S_{j_1}, i.e., the list $L(j_1)$, and from each list $L(k)$, $k \neq j_1$, delete the first element if it corresponds to a distance to a disk that has already been omitted. Proceeding in this way, after $O(n^2)$ time, we reach the stage where all distances between the remaining discrete disks are equal to zero. We now apply Theorem 4.6. Since we know that all remaining disks share a common node, we can apply the linear time algorithm in Kariv and Hakimi (1979) to identify such a node, say v_t. If we let $\{S_j\}$, $j \in J$ denote the subset of the remaining disks, then from Theorem 4.6 we conclude that the ordering of the elements $\{d(v_j, v_t) - s_j | j \in J\}$ induces the nest ordering of the remaining disks. Therefore we conclude that the total time to obtain the nest ordering of the rows of the $n \times n$ incidence matrix from the lists $\{L(j)\}$, $j = 1, ..., n$, is indeed $O(n^2)$.

We will now show how to compute all the lists $\{L(j)\}$, $j = 1, ..., n$, in $O(n^2)$ total time.

4.3.1 Computing the lists $\{L(j)\}$

The process is inductive on the size of the given tree. We use centroid decomposition as in Megiddo et al., (1981). In $O(n)$ time we find a centroid node, say v_k, of the given tree T. v_k decomposes the tree T into two subtrees $T^1 = (V^1, E^1)$ and $T^2 = (V^2, E^2)$, such that $V^1 \cap V^2 = \{v_k\}$, and $|V^t| \leq (2n + 1)/3$, $t = 1, 2$. Suppose, inductively, that for each $v_j \in V^1$ ($v_j \in V^2$), the list $L^1(j)$ ($L^2(j)$) of distances from S_j, the discrete disk centered at v_j, to all other disks centered at V^1 (V^2) has already been computed.

For each disk S_j, let $v_{k(j)}$ denote the closest node to the centroid v_k in S_j. Define

$$V_2^2 = \{v_t \in V^2 - \{v_k\} | v_{k(t)} \in V^2 - \{v_k\}\},$$

$$V_1^2 = V^2 - V_2^2 - \{v_k\},$$

$$V_1^1 = \{v_t \in V^1 - \{v_k\} | v_{k(t)} \in V^1 - \{v_k\}\},$$

$$V_2^1 = V^1 - V_1^1 - \{v_k\}.$$

The effort to compute the nodes $v_{k(j)}$, for $j = 1, ..., n$ is clearly $O(n^2)$.

We now describe how to construct all the lists $L(j)$, $v_j \in V^1$ in $O(n^2)$ total time. (A similar process can then be applied to construct the lists $L(j)$, $v_j \in V - V^1$.)

In $O(n \log n)$ time compute, and sort the distances $\{d(v_k, v_{k(t)}) | v_t \in V_2^2\}$. (Ties are partially resolved according to the ordering induced by the numbers $\{d(v_t, v_k) - s_t\}$, as required by Theorem 4.3.) Note that $d(v_k, v_{k(t)}) = d(v_k, S_t)$. Let L^* denote the sorted list. Next, in $O(n \log n)$ time compute and sort the numbers $\{d(v_t, v_k) - s_t | v_t \in V_1^2\}$. Let L' denote the sorted list.

Case I: Computing the list $L(k)$.

In this case the list $L(k)$ is attained by merging the lists $L^1(k)$ and $L^2(k)$.

Case II: Computing the lists $L(j)$ for all $v_j \in V_1^1$.

For each $v_j \in V_1^1$ we have $s_j < d(v_k, v_j)$. Let $L^*(j)$ denote the list obtained from L^* by adding the term $d(v_k, v_{k(j)})$ to each element in L^*. Traverse the path from v_k to $v_{k(j)}$, following the ordering induced by L', and compute the (sorted) list $L^{**}(j)$ of distances $\{d(S_j, S_t) = d(v_{k(j)}, S_t) | v_t \in V_1^2\}$. Note that at this point the set of distances $\{d(S_j, S_i) | i = 1, ..., n\}$ is partitioned between the three lists $L^1(j)$, $L^*(j)$, and $L^{**}(j)$. Merging these three sets yields the list $L(j)$. The time needed to compute $L(j)$, (when $L^1(j)$ is already given), is $O(n)$.

Case III: Computing the lists $L(j)$ for all $v_j \in V_2^1$.

For each $v_j \in V_2^1$ we have $s_j \geq d(v_k, v_j)$. Thus, $d(S_j, S_t) = 0$, for each $v_t \in V_1^2$. Let $L^{**}(j)$ be the list obtained from the set $\{d(S_j, S_t) | v_t \in V_1^2\}$, by ordering according to L'. Let $L^*(j)$ be the sorted list of the elements $\{d(S_j, S_t) | v_t \in V_2^2\}$. (Note that unlike **Case II**, the ordering of $L^*(j)$ is not necessarily idependent of j.) The list $L(j)$ is obtained by merging the three lists, $L^1(j)$, $L^{**}(j)$ and $L^*(j)$ in $O(n)$ time.

To show that the total time in **Case III** is also $O(n^2)$, we will now construct all the lists $L^*(j)$, $v_j \in V_2^1$ in $O(n^2)$ time. First we note that for each $v_j \in V_2^1$, and $v_t \in V_2^2$, $d(S_j, S_t) = d(S_j^k, v_{k(t)})$, where S_j^k is the discrete disk centered at v_k of radius $s'_j = s_j - d(v_k, v_j)$. Therefore, constructing the lists $L^*(j)$, $v_j \in V_2^1$ reduces to the following problem:

Problem 1

Given the subtree $T^2 = (V^2, E^2)$, with $|V^2| = n$, and a set of m discrete disks $S'_1, ..., S'_m$, all centered at some distinguished node $v_k \in V^2$, of radii $r_1, ..., r_m$, respectively, find in $O(n^2 + m)$ total time for each discrete disk S'_i, the sorted list of distances $\{d(S'_i, v_t) | v_t \in V^2\}$. (It is assumed that $r_1 < r_2 < ... < r_m$.) Since we deal with discrete disks, it is sufficient to deal with the case where each of the given radii is of the form $d(v_k, v_i)$, where $v_i \in V^2$.

We solve **Problem 1** by applying centroid decomposition on T^2. To facilitate the discussion, for each $v_i \in V^2$, let L_i be the sorted list of distances of the discrete disk of radius $r_i = d(v_k, v_i)$, centered at v_k, to all nodes $v_t \in V^2$. Let v_q be a centroid decomposing the tree T^2 into two subtrees $T^2(1) = (V^2(1), E^2(1))$ and $T^2(2) = (V^2(2), E^2(2))$, such that $V^2(1) \cap V^2(2) = \{v_q\}$, and $|V^2(u)| \leq (2n+1)/3$, $u = 1, 2$. Suppose without loss of generality that $v_k \in V^2(1)$. Let $L_i(1)$ be the sorted list of distances of the discrete disk S'_i of radius $r_i = d(v_k, v_i)$, centered at v_k, to all nodes $v_t \in V^2(1)$. Let $L_i(2)$ be the respective sorted list of distances from S'_i to the rest of the nodes in V^2.

First, inductively, we compute all the lists $\{L_i(1)\}$. Next we compute the list $L_i(2)$ for each index i such that $d(v_k, v_i) < d(v_k, v_q)$. Let L^* be the list of sorted distances obtained from the set $\{d(v_q, v_t) | v_t \notin V^2(1)\}$. L^* is computed in $O(n \log n)$ time. Now, for each disk S_i' with $d(v_k, v_i) < d(v_k, v_q)$, the list $L_i(2)$ is obtained from L^* by adding the constant $d(S_i', v_q)$ to each element in L^*. To obtain L_i, we then merge the lists $L_i(1)$ and $L_i(2)$ in linear time.

Let I be the index set of all disks S_i' such that $d(v_k, v_i) \geq d(v_k, v_q)$. To compute the lists $L_i(2)$, $i \in I$, we now inductively consider the subproblem of computing the distances to the nodes in $V^2(2)$ from the set of discrete disks, all centered at v_q, of radii $\{d(v_i, v_k) - d(v_k, v_q) | i \in I\}$. Again, for each $i \in I$, the list L_i is obtained by merging the lists $L_i(1)$, and $L_i(2)$.

To bound the total effort to solve **Problem 1**, denote by $K(n)$ the total effort needed to construct all the lists L_i, for a tree T^2 with n nodes . Then,

$$K(n) \leq cn^2 + K(n_1') + K(n_2'),$$

where c is constant, $n_1' + n_2' = n + 1$, $|V^2(1)| = n_1' \leq (2n + 1)/3$, and $|V^2(2)| = n_2' \leq (2n + 1)/3$. The above relation implies that $K(n) = O(n^2)$.

We summarize **Case III**, and note that the total time to construct all the lists $L(j)$, $v_j \in V_2^1$ is $O(n^2)$.

So far we have shown how to construct all the lists $L(j)$, $j = 1, ..., n$, from the lists $L^1(j)$, $L^2(j)$, $j = 1, ..., n$, in $O(n^2)$ time. To calculate the total effort (including the time to compute the lists $L^1(j)$, $L^2(j)$, $j = 1, ..., n$), involved in the above inductive process, let $n_1 = |V^1|$ and $n_2 = |V^2|$. Denote by $C(n)$ the total effort needed to construct all the lists $L(j)$, for a tree T with n nodes $\{v_1, ..., v_n\}$. Then,

$$C(n) \leq cn^2 + C(n_1) + C(n_2),$$

where c is constant, $n_1 + n_2 = n + 1$, $n_1 \leq (2n + 1)/3$, and $n_2 \leq (2n + 1)/3$. The above relation implies that $C(n) = O(n^2)$.

We can now conclude that the total time needed to find a nest ordering of the rows of the matrix of two collections of discrete disks is quadratic.

Following Kolen and Tamir (1990), the total time to convert such a matrix to greedy form is therefore $O(n^2)$. From Hoffman et al., (1985) it follows that the weighted covering problem defined by such a matrix is also solvable in $O(n^2)$ time.

Remark 4.7 *In Megiddo et al. (1981) the centroid decomposition approach is used to generate in $O(n \log^2 n)$ time, an implicit representation of the set $\{d(v_i, v_j)|i, j = 1, ..., n\}$. This representation allows efficient (subquadratic) selection in the set. We conjecture that the above centroid decomposition process to generate the lists $\{L(j)\}$ can be sped up to yield a similar representation of the distances $\{d(S_i, S_j)|i, j = 1, ..., n\}$ in $O(n \log^2 n)$ time. This may lead to a subquadratic algorithm for finding the nest ordering.*

4.4 Path Graphs

The above discussion simplifies significantly when the tree is a path. In this case each truncated disk is a subpath. A truncated disk can be represented as a continuous disk. For example, if $P(v_k, v_t)$ is a path, we can view it as a continuous disk of radius $r = d(v_k, v_t)/2$, centered at the midpoint of the path. With this observation we can now apply the transformation in Theorem 3.1 to represent the incidence matrix of two collections of paths on a path graph as the incidence matrix of disks (rows) vs. nodes (columns) in a simple tree obtained from the path by extending additional edges from some points of the underlying path graph. Specifically, the related unweighted covering problem can be solved in linear time on this tree by the algorithm of Kariv and Hakimi (1979). The weighted covering problem is solved in $O(n \log n)$ time by the algorithm in Bertossi and Gori (1988). If the endpoints of the intervals are already sorted, then the weighted covering problem can be solved in $O(n)$ time by the algorithm in Chang (1998). Due to the structure of the sets R_1 and R_2 we can solve both the discrete and the continuous generalized p-center problems in $O(n \log n)$ time.

4.5 Trubin Matrices

In this section we consider another class of totally balanced matrices defined by a collection of subtrees, for which the nest ordering, and therefore the greedy form, can be obtained in $O(n^2)$ time. Let $T = (V, E)$ be a tree with $V = \{v_1, ..., v_n\}$ and $E = \{e_2, ..., e_n\}$. Following Trubin (1983), for each $j = 1, ..., n$, define the set of subtrees $T^*(j, k)$, $k = 1, ..., n$, recursively by:

$T^*(j, 1)$ consists of the single node v_j only.

For $k = 2, ..., n$, $T^*(j, k)$ is obtained from $T^*(j, k-1)$ by augmenting the unique edge (and its two nodes), which has the smallest index amongst all edges that are incident to $T^*(j, k-1)$, i.e., have exactly one node in $T^*(j, k-1)$. v_j is called the *root* of $T^*(j, k)$. We note that for $k = 2, ..., n$, $T^*(j, k)$ contains $T^*(j, k-1)$, i.e., the latter is a subtree of the former.

Consider the Trubin family of subtrees $\{T^*(j, k) | j, k = 1, ..., n\}$. For each node v_i, and subtree $T^*(j, k)$, define $a_{i,(j,k)}$ to be equal to 1 if v_i is in $T^*(j, k)$ and 0 otherwise. Let $A = (a_{i,(j,k)})$ be the respective incidence matrix with n rows and n^2 columns. It follows from the discussion in Trubin (1983) that A is totally balanced. We call such a matrix a *Trubin matrix.* We will present a simple proof of this result by identifying explicitly a pair of leaf nodes of T, (rows of A), each one of them possessing the nestedness property with respect to the above family of subtrees. This proof will also lead to an $O(n^2)$ algorithm to find a nest ordering of the rows of a Trubin matrix.

We define these leaf nodes inductively. First, for each subtree T^* with at least two nodes, define the edge of T^* with the largest index to be the *heavy* edge of T^*. If T consists of a single node declare it to be *simplicial.* If T consists of a single edge declare both of its nodes to be simplicial nodes. Suppose that $T = (V, E)$ has $n > 2$ nodes, and $E = \{e_2, ..., e_n\}$. If $e_n = (v_i, v_j)$, let T' and T" be the two subtrees obtained by cutting e_n, where v_i is in T', v_j is in T", and T' has at least two nodes. e_n is the the heavy edge of

T. Let e' be the heavy edge of T'. (e' is the edge of T' with the largest index.) Then it is easy to see that there is one simplicial node v' of T', such that e' is on the unique simple path in T' connecting v_i with v'. If T" has a single node, v_j, then declare v' and v_j to be the pair of simplicial nodes associated with T. Otherwise, let e" be the heavy edge of T". Then there is a simplicial node v" of T", such that e" is on the unique simple path connecting v" with v_j. Declare v' and v" to be a pair of simplicial nodes of T.

Theorem 4.8 *Let $T = (V, E)$ be a tree, and let A be the respective Trubin matrix. Let v_p, v_q, be a pair of simplicial (leaf) nodes of T, defined above. Then the rows of A corresponding to v_p, v_q are simplicial.*

Proof: Consider a simplicial leaf node v_p. Suppose that $e_n = (v_i, v_j)$, where v_j is on the path connecting v_i with v_p. From the inductive definition of the simplicial nodes it follows that there is a sequence of edges $e_{j(1)}, ..., e_{j(t)}$, $j(t) = n$, on the path connecting v_i with v_p, with the following properties:

(1) $j(1) < j(2)... < j(t)$.

(2) v_p is a leaf of $e_{j(1)}$.

(3) For each $m = 1, ..., t$, let T_m be the connected component of T, which contains exactly one node of $e_{j(m)}$ and one node of $e_{j(m+1)}$, and is obtained by cutting the edges $e_{j(m)}$ and $e_{j(m+1)}$. (For convenience, $e_{j(t+1)}$ is defined as the other simplicial node of T, v_q.) If e_i is an edge of T_m, then $i < j(m)$.

To prove the nestedness property consider a pair of subtrees $T^*(j_1, k_1)$ and $T^*(j_2, k_2)$ in the Trubin family defined above, and assume that both contain the simplicial node v_p. Suppose without loss of generality that v_{j_1} (v_{j_2}), the root of $T^*(j_1, k_1)$ ($T^*(j_2, k_2)$) is in T_{m_1} (T_{m_2}), with $m_1 < m_2$.

From the definition of the family, and the above properties, it now follows that $T^*(j_1, k_1)$ contains $T_1 \cup ... \cup T_{m_1}$, and $T^*(j_2, k_2)$ contains $T_1 \cup ... \cup T_{m_2}$. If $T^*(j_1, k_1)$ does not intersect e_{m_2+1}, then clearly it is contained in $T^*(j_2, k_2)$. Otherwise, both $T^*(j_1, k_1)$ and $T^*(j_2, k_2)$, coincide with two subtrees in the family that are rooted at the same node, i.e., a node of e_{m_2+1}. Hence, one of these subtrees must contain the other one. This completes the proof. ■

Remark 4.9 *Note that a leaf edge of maximum index does not necessarily correspond to a simplicial node, as illustrated by the example of the tree with six nodes and* $e_2 = (v_1, v_2), e_3 = (v_4, v_5), e_4 = (v_3, v_6), e_5 = (v_3, v_4)$, *and* $e_6 = (v_2, v_3)$. *Also, a leaf edge of minimum index does not necessarily correspond to a simplicial node, as illustrated by the example of the tree with four nodes and* $e_2 = (v_1, v_2), e_3 = (v_1, v_3)$, *and* $e_4 = (v_1, v_4)$.

The above theorem can be used to find a permutation leading to the nest ordering of the nodes (rows) of a Trubin matrix in $O(n^2)$ time. It is sufficient to show that the largest indexed edge e_n, and a pair of simplicial nodes, v', v", of a given tree T, with e_n on $P(v', v")$, can be found in $O(n)$ time. We apply a result in Chazelle (1987), that after $O(n)$ preprocessing time, for any pair of nodes v_k, v_t, and an edge e_i, it takes constant time to determine whether e_i is on $P(v_k, v_t)$. From the definition of the pair of simplicial nodes, given above, Using Chazelle (1987), it now follows that v', v" can be obtained from the respective pairs of the subtrees T' and T" in constant time. Argueing inductively we conclude that v' and v" can be found in $O(n)$ time.

5 Concluding Remarks

In the above generalized p-center problems it is implicitly assumed that the cost of establishing a center (server) is fixed and independent of its location. We can further extend this model and include setup costs which depend on the location. Specifically, suppose that the cost of establishing a center at node v_j, $j = 1, ..., n$, is a_j. In the *generalized budgeted center problem* there is a total budget of B for setting up centers at the nodes of the tree, and the objective is to minimize the maximum response time, without violating the budget constraint. This model can be solved efficiently by the approach used above for the p-center version. The only difference is that for the budgeted model, we need to solve weighted covering problems. From the above we conclude that the complexity of solving the generalized budgeted center problems is $O(n^2 \log n)$.

In the above models customers and servers are allowed to meet either at any point of the network (the continuous model), or at any node (the discrete model). We may consider a third variant, where a customer v_i and a server located at v_j can meet either at v_i or at v_j, but nowhere else. In other words, a customer v_i can be served only at its own location, or at any node where a server is located. (In the latter case the service will definitely be provided at the location of the closest server to v_i.) We call this version of the model, the *binary* p-center problem. It should be noted that even the case of a path graph is not known to be efficiently solvable. The incidence matrices defining the covering problems are not necessarily totally balanced. For a given response time t, the elements of the incidence matrix $A = (a_{i,j})$ are defined by $a_{i,j} = 1$ if and only if $d(v_i, v_j) \leq t \max\{w_i, w'_j\}$.

Example 5.1 *Consider the following set of nodes (points on the real line):* $v_1 = 0, v_2 = 2, v_3 = 5, v_4 = 6, v_5 = 7$, *and* $v_6 = 12$. *Suppose that* $w_i = w'_i$, $i = 1, ..., 6$, *and* $w_1 = 6, w_2 = 0, w_3 = 5, w_4 = 0, w_5 = 5$, *and* $w_6 = 6$. *For* $t = 1$, *the submatrix defined by the rows* $\{1, 2, 6\}$ *and the columns* $\{3, 4, 5\}$, *satisfy* $a_{1,3} = a_{1,4} = 1$, $a_{1,5} = 0$; $a_{2,3} = 1$, $a_{2,4} = 0$, $a_{2,5} = 1$; *and* $a_{6,3} = 0$, $a_{6,4} = a_{6,5} = 1$. *This submatrix is neither totally balanced nor totally unimodular.*

The three versions of the generalized p-center problem, the continuous, the discrete and the binary versions, are all NP-hard when the underlying graph is general, and not necessarily a tree. In a separate paper we will discuss approximations algorithms for the three versions defined on general undirected graphs.

References

[1] A.A. Bertossi and A. Gori, "Total domination and irredundance in weighted interval graphs," *SIAM J. on Discrete Mathematics* **1**, (1988), 317-327.

[2] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, and R.E. Tarjan, "Time bounds for selection," *J. Compute. System Science* **7**, (1972), 448-461.

[3] K.S. Booth and J.H. Johnson, "Dominating sets in chordal graphs," *SIAM J. on Computing* **11**, (1982), 191-199.

[4] M.W. Broin and T.J. Lowe, "A dynamic programming algorithm for covering problems with (greedy) totally balanced constraint matrices," *SIAM J. on Algebraic and Discrete Methods* **7**, (1986), 348-357.

[5] M.-S. Chang, "Efficient algorithms for the domination problems on interval and circular-arc graphs," *SIAM Journal on Computing* **27**, (1998), 1671-1694.

[6] B. Chazelle, "Computing on a free tree via complexity-preserving mappings," *Algorithmica* **2**, (1987), 337-361.

[7] S.W. Cheng, M. Kaminski and S. Zaks, "Minimum dominating sets of intervals on lines," *Algorithmica* **20**, (1998), 294-308.

[8] G.N. Frederickson and D.B. Johnson, "Finding k-th paths and p-centers by generating and searching good data structures," *J. of Algorithms* **4**, (1983), 61-80.

[9] A.J. Hoffman, A. Kolen and M. Sakarovitch, "Totally balanced and greedy matrices," *SIAM J. on Algebraic and Discrete Methods* **6**, (1985), 721-730.

[10] O. Kariv and S.L. Hakimi, "An algorithmic approach to network location problems: Part I. The p-centers," *SIAM J. on Applied Mathematics* **37**, (1979), 513-538.

[11] A. Kolen, "Solving covering problems and the uncapacitated plant location problem on trees," *European J. of Operational Research* **12**, (1983), 266-278.

[12] A. Kolen and A. Tamir, "Covering problems," in P.B. Mirchandani and R.L. Francis, eds. Discrete Location Theory, Wiley, New York, (1990).

[13] A. Lubiw, "Doubly lexical ordering of matrices," *SIAM J. on Computing* **16**, (1987), 854-879.

[14] N. Megiddo and A. Tamir, "New results on the complexity of p-center problems," *SIAM J. on Computing* **12**, (1983), 751-758.

[15] N. Megiddo, A. Tamir, E. Zemel and R. Chandrasekaran, "An $O(n\log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems," *SIAM J. on Computing* **10**, (1981), 328-337.

[16] R. Paige and R.E. Tarjan, "Three partition refinement algorithms," *SIAM J. on Computing* **16**, (1987), 973-989.

[17] A. Tamir "A class of balanced matrices arising from location problems," *SIAM J. on Algebraic and Discrete Methods* **4**, (1983), 363-370.

[18] A. Tamir, "Totally balanced and totally unimodular matrices defined by center location problems," *Discrete Applied Mathematics* **16**, (1987), 245-263.

[19] V.A. Trubin, "Two classes of location problems on tree networks," *Cybernetics* **19**, (1983), 539-544.