Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 156 (2008) 2890–2910
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Center location problems on tree graphs with subtree-shaped customers

J. Puerto[a], A. Tamir[b,*], J.A. Mesa[c], D. Pérez-Brito[d]

[a] *Facultad de Matemáticas, Universidad de Sevilla, Spain*
[b] *School of Mathematical Sciences, Tel Aviv University, Israel*
[c] *ETS Ingenieros, Universidad de Sevilla, Spain*
[d] *ETS Ingeniería Informática, Universidad de La Laguna, Spain*

Received 4 April 2006; received in revised form 16 September 2007; accepted 27 November 2007
Available online 21 February 2008

Abstract

We consider the p-center problem on tree graphs where the customers are modeled as continua subtrees. We address unweighted and weighted models as well as distances with and without addends. We prove that a relatively simple modification of Handler's classical linear time algorithms for unweighted 1- and 2-center problems with respect to point customers, linearly solves the unweighted 1- and 2-center problems with addends of the above subtree customer model. We also develop polynomial time algorithms for the p-center problems based on solving covering problems and searching over special domains.

Keywords: Facility location; Subtree-shaped customers; Tree graphs

1. Introduction

In a typical center problem defined on a network (a metric space induced by an undirected graph $G = (V, E)$ and its positive edge lengths), there is a set of demand points (customers), represented by a subset of points of the network. Each demand point is associated with a cost function, transportation or service cost, which is assumed to be a nondecreasing linear function of the distance of the point to its server. Given an integer p, the objective is to locate p servers (facilities) on the network, minimizing the maximum transportation cost of the customers. (Servers are assumed to be uncapacitated and identical in terms of their services, so that each demand point is served by its respective closest server.) The above model is called the p-center problem. The seminal paper of Hakimi [16], which is considered as a milestone in modern location theory, provides an algorithmic framework to deal with single server problems. An extended algorithmic framework to multifacility center location problem on networks is presented in the paper by Kariv and Hakimi [22]. In that paper the authors consider problems where the set of demand points is finite and coincides with the node set of the underlying graph. It is shown that the multifacility problems are NP-hard. Moreover, polynomial time algorithms are presented for the p-center problem defined on tree networks.

* Corresponding author.
E-mail address: atamir@post.tau.ac.il (A. Tamir).

0166-218X/$ - see front matter
doi:10.1016/j.dam.2007.11.022

The reader is referred to Megiddo et al. [30], Frederickson and Johnson [12], Megiddo and Tamir [29], Cole [7] and Frederickson [11] for the most efficient algorithms, known today, for various versions of the p-center problem on tree networks when the demand set is finite. The first polynomial time algorithm for the p-center problem when the demand set is the continuum set of all points on a tree network is given in Chandrasekaran and Daughety [5]. This version is called the continuous p-center problem. Generalizations are discussed and polynomially solved in Tamir and Zemel [36]. The reader is referred to Frederickson and Johnson [12], Megiddo and Tamir [29] and Cole [7] for the most efficient algorithms, known today, for the continuous p-center problem on tree networks. In the last decade there has been a great interest in locating "extensive" servers (new facilities), like paths or subtrees of a network, which cannot be represented as points of the network or points in a continuous space, (see the surveys by Plastria [32] and Díaz-Báñez et al., [8]). The interested reader is referred to [33] and the references therein for a variety of the most recent algorithmic and structural results. In this paper we study and focus on location problems with "extensive" customers, (see the paper by Nickel et al., [31] and the references therein). For motivation purposes consider a network with communication lines, where each line is used exclusively to connect between a pair of points. To check whether a line properly functions it is sufficient to reach any point on that line, and transmit to each one of its endpoints. (From a given point the server can test the functionality of all lines passing through that point.) In this model each communication line is viewed as an extensive customer on the network and the servers are the checkpoints that we need to choose on the network so that we cover all the extensive customers (lines). The optimization problem is to find the minimum number of checkpoints needed to cover all lines.

Another example of such a model is the round-trip problem studied by Chan and Francis [6], Kolen [23], Kolen and Tamir [24] and Tamir and Halman [35]. In this model a customer is represented by some fixed point on the network. The customer is also associated with its depot, e.g., a fixed warehouse, or a dumping site. When a call is placed by a customer, a server is required to depart from its home base, travel to the depot (customer), deliver some good from the depot (customer) to the customer (depot), and return to its home base. The transportation cost is assumed to be a linear function of the distance of the round-trip traveled by the server. More general models are studied by Berman, Simchi-Levi and Tamir [3]. (Here the extensive demand facilities are the shortest paths connecting each point customer with its respective depot. For further details the reader is referred to Section 6.) In all the above examples subtrees represent the extensive customers. To motivate the fact that they may be non-disjoint, consider the first example. Two communication lines, each one connecting a different pair of two terminals, may in practice pass through some common intermediate points. Such points can be used as potential checkpoints for both lines.

Motivated by the above examples of extensive customers, we consider center problems on tree graphs where the demand originates at continua (extensive) connected facilities: paths, neighborhoods and general subtrees. As in Berman, Simchi-Levi and Tamir [3], we can view the extensive customer model as follows:

There is a finite collection of subsets S_i, $i \in I = \{1, \ldots, m\}$, where $S_i \subseteq V$. S_i is viewed as a set of demand points. (Let $n = |V|$.) When a demand is originated at some subset S_i, a server located at x must travel and visit each of the customers in S_i and return to its home base. The total travel distance of the server will be $2d(x, T_i) + 2L(T_i)$, where T_i is the subtree induced by S_i, $d(x, T_i)$ is the distance between x and its closest point in T_i, and $L(T_i)$ is the length of T_i. The transportation cost of S_i is assumed to be a linear function of $2d(x, T_i) + 2L(T_i)$. Specifically, for $i \in I$, let w_i and k_i be a pair of reals with $w_i \geq 0$. Then the transportation cost is $w_i(d(x, T_i) + k_i)$. (The terms k_i and w_i are referred to as the addend and weight of the extensive customer T_i, respectively. Due to the form of the distance function which depends on T_i we refer to the extensive T_i customer instead of S_i customer.) The p-center problem of the above extensive customer model is to locate p points (servers) on the tree network, minimizing the maximum over all transportation costs of the m extensive customers to their respective nearest servers. By the unweighted version we refer to the case where $w_i = 1$ for all $i \in I$, while the model without addends corresponds to the case where $k_i = 0$ for all $i \in I$.

We present polynomial time algorithms to solve the above p-center problems. These algorithms are based on solving covering problems and searching over specially structured domains. Our algorithmic results are summarized in Tables 1 and 2.

In addition to the general algorithmic results, we focus on the case, relevant to most applications, where the number of servers p is significantly smaller than the number of customers. In particular, we prove that the classical linear time algorithms of Handler (1973, 1978) [18,19] for the unweighted 1- and 2-center problems with respect to point customers with no addends extend to the unweighted model with extensive subtree customers with addends, while keeping the linear complexity. In the single facility case Handler considers the problem of finding a point on a tree

Table 1
Resume of the results in the paper

Unweighted models					
Structure	+addends	Problem			
		1-center	2-center	Covering	p-center
Points	Without	$O(n)$ [18]	$O(n)$[19]	$O(n)$[34]	$O(n)$ [10]
	With	$O(n)$ [25]	$O(n)$ [25]	$O(n)$ [34]	$O(n \log n)$ [12]
Discrete	Without	$O(m+n)$	$O(m+n)$	$O(n+p\log n+pm)$[c]	$O(mn\log(n))$[b]
paths	With	$O(n+m)$	$O(n+m)$	$O(n+p\log n+pm)$[c]	$O(mn\log(mn))$
Neighborhoods	Without	$O(m+n)$	$O(m+n)$	$O(m+n)$ [34]	$O((m+n)\log n)$
	With	$O(m+n)$	$O(m+n)$	$O(m+n)$	$O((m+n)\log(mn))$
Discrete	Without	$O(n+t)$[a]	$O(n+t)$	$O(\min\{mn, n+p\log n+pt\})$[c]	$O(mn\log n)$
subtrees	With	$O(n+t)$	$O(n+t)$	$O(\min\{mn, n+p\log n+pt\})$[c]	$O(mn\log(mn))$ [35]

[a] t is the total number of leaves of the m subtrees in $\mathcal{T}$.
[b] The optimal value is the maximum of the addends or an element in the set $\{\frac{d(v_s,v_t)+k_i+k_j}{2}\}_{v_s,v_t\in V,\ i,j=1,\ldots,m}$.
[c] Time for deciding the existence of a p covering.

Table 2
Resume of the results in the paper

Weighted models				
Structure	+addends	Problem		
		1-center	Covering	p-center
Points	Without	$O(n)$ [28]	$O(n)$ [34]	$O(n\log^2 n)$ [30]
$\lvert I\rvert = n$	With	$O(n)$ [28]	$O(n)$ [34]	$O(n\log^2 n)$ [30]
Discrete	Without	$O(n+m\log n)$	$O(n+p\log n+pm)$[a]	$O(mn\log^2(mn))$
paths	With	$O(n+m\log n)$	$O(n+p\log n+pm)$[a]	$O(\min[mn\log^2(mn), (p\log m+\log n\log m)(n+p\log n+pm)])$
Neighborhoods	Without	$O(m+n)$	$O(m+n)$	$O((m+n)\log^2(m+n))$
	With	$O(m+n)$ [28]	$O(m+n)$	$O((m+n)\log^2(m+n))$
Discrete	Without	$O(n+t\log n)$	$O(\min\{mn, n+p\log n+pt\})$[a]	$O((mn)\log^2(mn))$
subtrees	With	$O(n+t\log n)$	$O(\min\{mn, n+p\log n+pt\})$[a]	$O(\min[(mn)\log^2(mn), (p\log t+\log m\log n)(n+p\log n+pt)])$

[a] Time for deciding the existence of a p covering.

network minimizing the (unweighted) maximum distance to the nodes of the tree. (The problem is equivalent to finding the diameter of the tree.) His classical solution is obtained as follows: Choose an arbitrary point y. Find a node, say v_j, which is furthest away from y. Find a node, say v_k, which is furthest away from v_j. The path connecting v_j and v_k is a diameter of the tree, and its midpoint, say x_1, is the unique solution to the unweighted 1-center problem. (We note that in a recent paper, Bulterman et al. (2002) [4], attribute this algorithm to Dijkstra.) Halfin (1974) [17] and Lin (1975) [25] extended Handler's algorithm to the unweighted 1-center problem with addends.

To solve the 2-center problem, following Handler [19], split the tree into two subtrees, say T^j and T^k, containing v_j and v_k respectively, and satisfying $T^j \cup T^k = T$, $T^j \cap T^k = \{x_1\}$. Let x^j and x^k be the 1-centers of T^j and T^k respectively. Then (x^j, x^k) solve the 2-center problem on T. (Handler's algorithm for the 2-center problem was recently rediscovered by Huang et al. [21].)

We prove that the above algorithms are valid for a general collection $\{T_i\}$ of extensive subtree customers. Specifically, to solve the unweighted 1-center problem without addends, choose an arbitrary point y, which is not in $\cap_{i\in I}\{T_i\}$. Find a subtree, say T_j, which is furthest away from y. Find a subtree, say T_k, which is furthest away from T_j. Then the midpoint of the path connecting T_j and T_k, say x_1, is the unique solution to the unweighted 1-center problem. (The problems with addends are slightly more complex.)

The organization of the paper is as follows: In Section 2 we present preliminary results which are instrumental in the rest of the paper. In particular, we develop simple expressions to compute distances between pairs of paths, neighborhoods, and general subtrees. Section 3 is devoted to the unweighted 1-center problem with addends of a collection of subtrees. We present a linear time algorithm which extends the intuitive ideas and the classical algorithm of Handler [18] for the standard 1-center problem for point customers. In the following section we consider the unweighted 2-center problem with addends of a collection of subtrees. Again, we present a linear time algorithm that resembles the approach followed by Handler [19] in his classical algorithm for the unweighted 2-center problem. Section 5 deals with the weighted 1-center problem with addends of a collection of subtrees. We provide a subquadratic algorithm solving the general case. Section 6 covers the weighted p-center problem. We develop several algorithms that are based on solving the covering problem and then using efficient search on structured domains or applying the parametric approach. Finally, in Section 7 we make some final comments and pose a few open problems related to the ones considered in the paper.

2. Notation, definitions and preliminaries

Let $T = (V, E)$ be an undirected tree network with node set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{e_2, \ldots, e_n\}$. Each edge e_j, $j = 2, 3, \ldots, n$, 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. Each subgraph of T is also viewed as a subset of $A(T)$. We refer to an interior point on an edge by its Euclidean distances along the edge to the nodes of the edge. Let $P_{ij} = P[v_i, v_j]$ denote the unique simple path in $A(T)$ connecting v_i and v_j. $d(v_i, v_j)$ will denote the length of $P[v_i, v_j]$. Suppose that the tree T is rooted at some distinguished node, say v_1. For each node v_j, $j = 2, 3, \ldots, n$, let $v_{p(j)}$, 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 then the child of $v_{p(j)}$. v_j is a leaf if it has no children. A node v_i is a descendant of v_j if $v_j \in P[v_1, v_i]$. v_j is then an ancestor of v_i. Define $level(v_i) = depth(v_i)$ to be the number of nodes on $P[v_i, v_1]$. A Level Ancestor Query ($LA(v_i; k)$) is to find the depth k ancestor of v_i. Given a pair of nodes v_i, v_j, their least common ancestor ($LCA(v_i, v_j)$) is a node of maximum depth, which is an ancestor of both v_i and v_j. The edge lengths induce a distance function on $A(T)$. For any pair of points $x, y \in A(T)$, let $d(x, y)$ denote the length of $P[x, y]$, the unique simple path in $A(T)$ connecting x and y. With this distance function $A(T)$ is viewed as a metric space. For any pair of compact subsets $X, Y \subseteq A(T)$, $d(X, Y) = \min\{d(x, y) : x \in X, y \in Y\}$. A subtree is a compact and connected subset of $A(T)$. A subtree is discrete if its relative boundary points are nodes. Note that for any pair of subtrees $X, Y \subseteq A(T)$, and a point $z \in A(T)$, $d(X, Y) \leq d(X, z) + d(Y, z)$. If $d(X, Y) > 0$, there are unique points $x' \in X$ and $y' \in Y$ such that $d(X, Y) = d(x', y')$. In this case we define $P[X, Y] = P[x', y']$. If Y is a singleton and $d(X, Y) = 0$ we define $P[X, Y] = Y$. Given a point $x \in A(T)$ and a nonnegative real r, $N_x(r)$, the neighborhood subtree of radius r centered at x, is defined by $N_x(r) = \{y \in A(T) : d(x, y) \leq r\}$. For any subtree T', and a node v_i of T', define $F(T' : v_i)$ to be the forest of T' obtained from T' by removing v_i and all edges of T' incident to v_i.

In this paper we consider the following multi-center location problem. Given a collection of m *discrete* subtrees $\mathcal{T} = \{T_i\}$, $i \in I = \{1, 2, \ldots, m\}$, a set of real numbers, $\{k_i\}$, $i \in I$, called addends, and nonnegative weights $\{w_i\}_{i \in I}$, the weighted p-center problem with addends is to find a set of p points, $X_p = \{x_1, \ldots, x_p\}$, $X_p \subseteq A(T)$, minimizing

$$\max_{i \in I}\{w_i(d(T_i, X_p) + k_i)\}.$$

We let r_p^* denote the optimal objective value. Since each subtree T_i is discrete, we may assume that $p < n$, and $p < m$. When $w_i = 1$ for all $i \in I$, we refer to the unweighted center problem, and when $k_i = 0$ for all $i \in I$, we refer to the center problem without addends.

In this section we present some results and their algorithmic implications involving collections of subtrees. We will later use these results to develop our efficient algorithms to solve the above p-center problem. Given the above collection of m *discrete* subtrees $\mathcal{T}$, each subtree T_i, $i \in I$ is represented in terms of its root x_i and its set of leaves, L_i^*. Set $n_i^l = |L_i^*| + 1$. x_i, the root of T_i, is the closest point of T_i to v_1.

Given a subset $I' \subseteq I$, a pair of subtrees T_r, T_s, $r, s \in I'$, is called a *diametrical pair* of I', if

$$d(T_r, T_s) + k_r + k_s = \max_{i,j \in I'}\{d(T_i, T_j) + k_i + k_j\}.$$

Note that r and s are not necessarily distinct. Moreover, if $d(T_r, T_s) = 0$ then both (T_r, T_r) and (T_s, T_s) are diametrical pairs. If $d(T_r, T_s) > 0$, for any point $x \in P[T_r, T_s]$, define a *diametrical partition* of T with respect to T_r, T_s and x by a pair of subtrees X, Y in $A(T)$ satisfying, $X \cup Y = A(T)$, $X \cap Y = \{x\}$, $T_r \subseteq X$ and $T_s \subseteq Y$.

Lemma 2.1. *Let x_q, $q \in I$ be such that there is no x_j, $x_j \neq x_q$ and $x_q \in P[x_j, v_1]$. Then $x_q \in T_i$ for all $i \in I$, or there exists some T_p such that $T_p \bigcap T_q = \emptyset$.*

Proof. Suppose $T_q \bigcap T_i \neq \emptyset$ for all $i \in I$. If $x_i = x_q$, then $x_q \in T_i$. If $x_i \neq x_q$, then by the minimality property of x_q, x_i is not in T_q. But since $T_i \bigcap T_q \neq \emptyset$, T_i contains some point y in T_q. Hence, from the connectivity of T_q, $P[y, x_i] \subseteq T_i$. But $x_q \in P[y, x_i]$, and therefore $x_q \in T_i$. □

By definition of x_q we obtain the next lemma.

Lemma 2.2. *Assume that there exists a diametrical pair $T_1,\ T_2 \in \mathcal{T}$ such that $d(T_1, T_2) + k_1 + k_2 > 2 \max_{i \in I} k_i$. Let x_q, $q \in I$ be such that there is no x_j, $x_j \neq x_q$ and $x_q \in P[x_j, v_1]$. If the collection $\mathcal{T}$ does not intersect at x_q, then T_q does not intersect the interior of $P[T_1, T_2]$.*

Given the rooted tree T, Harel and Tarjan (1984) [20] present data structures which require $O(n)$ preprocessing time, and enables one to find the nearest or least common ancestor (*LCA*) of a given pair of nodes or points in $A(T)$ in constant time. In addition, Dietz (1991) [9], Berkman and Vishkin (1994) [2], and Bender and Farach-Colton (2002) [1] present data structures which require $O(n)$ preprocessing time, and allows one to answer a level ancestor query in constant time. (The reader is referred to Bender and Farach-Colton (2002) [1] for the simplest data structure.) These results are very useful to evaluate distances between subtrees as illustrated in the next lemma.

Lemma 2.3. *Let x, y be a pair of points in $A(T)$, and let z be their least common ancestor. Then $d(x, y) = d(x, v_1) + d(y, v_1) - 2d(z, v_1)$, and $d(x, y)$ can be computed in constant time. Let x, y, z, u be four points in $A(T)$, then*

$$d(P[x, y], P[z, u]) = \max\{0, 1/2(d(x, z) + d(y, u) - d(x, y) - d(z, u))\},$$

and $d(P[x, y], P[z, u])$ can be computed in constant time. Let x, y be a pair of points in $A(T)$, and let r, s be a pair of nonnegative reals. Then

$$d(N_x(r), N_y(s)) = \max\{0, d(x, y) - r - s\}.$$

Let T_i and T_j be a pair of subtrees in $\mathcal{T}$. Let z be the least common ancestor of x_i and x_j, the roots of T_i and T_j respectively. Suppose that $x_i \neq x_j$. If $z \neq x_i$ and $z \neq x_j$, then $d(T_i, T_j) = d(x_i, x_j)$. If $z = x_j$, then

$$d(T_i, T_j) = d(T_j, x_i) = \min_{v_k \in L_j^*} \{d(x_i, P[x_j, v_k])\}.$$

In particular, $d(T_i, T_j)$ can be computed in $O(n_j^l)$ time. Let x be a point in $A(T)$. Then the set $\{d(x, T_j)\}$, $j \in I$, can be computed in $O(\sum_{j \in I} n_j^l)$ time. Let T_i be a subtree in $\mathcal{T}$. Then the set $\{d(T_i, T_j)\}$, $j \in I$, can be computed in $O(n + \sum_{j \in I} n_j^l)$ time.

Proof. The results about computing the distances between a pair of points, a pair of paths, a pair of neighborhoods, and a pair of subtrees in $\mathcal{T}$ are clear. Consider the computation of the set $\{d(x, T_j)\}$, $j \in I$ where $x \in A(T)$. For each $j \in I$, let z_j be the least common ancestor of x and x_j, the root of T_j. If $z_j \neq x$, and $z_j \neq x_j$, then $d(x, T_j) = d(x, x_j)$. If $z_j = x$, $d(x, T_j) = d(x, x_j)$, and if $z_j = x_j$

$$d(T_j, x) = \min_{v_k \in L_j^*} \{d(x, P[x_j, v_k])\}.$$

Therefore the set $\{d(x, T_j)\}$, $j \in I$, can be computed in $O(\sum_{j \in I} n_j^l)$ time. Finally, consider the last statement where T_i is an arbitrary subtree in $\mathcal{T}$. For each subtree $T_j \in \mathcal{T}$, we compute $z_{i,j}$, the least common ancestor of x_i and x_j. If $z_{i,j} \neq x_i$, and $z_{i,j} \neq x_j$, then $d(T_i, T_j) = d(x_i, x_j)$. Let $I_i^+ = \{j \in I : z_{i,j} = x_i\}$ and let $I_i^- = \{j \in I : z_{i,j} = x_j\}$. We clearly have, $d(T_i, T_j) = d(x_j, T_i)$, for $j \in I_i^+$, and $d(T_i, T_j) = d(x_i, T_j)$, for $j \in I_i^-$. By scanning the subtree of all descendants of x_i, starting at x_i, we can compute $d(x_j, T_i)$ for all $j \in I_i^+$ in $O(m + n)$ time. (Note that the time

to compute $\max_{j\in I_i^+}\{d(x_j, T_i)\}$ is only $O(n)$ since we consider discrete subtrees, and therefore there are only $O(n)$ different values of x_j, $j \in I_i^+$.) Next, consider a subtree T_j such that x_i is a descendant of x_j, i.e., $j \in I_i^-$. Then,

$$d(T_i, T_j) = d(T_j, x_i) = \min_{v_k\in L_j^*}\{d(x_i, P[x_j, v_k])\}.$$

In particular, $d(T_i, T_j)$ can be computed in $O(n_j^l)$ time. Hence, for $T_i \in \mathcal{T}$, the set $\{d(T_i, T_j)\}$, $j \in I$, can be computed in $O(n + \sum_{j\in I} n_j^l)$ time. □

To conclude, when all subtrees are discrete, and represented as above, (root and leaves), the time to compute distances from a given subtree T_i to all other subtrees is $O(n + \sum_{j\in I} n_j^l)$. This is a linear time algorithm in terms of the input size. In particular, for a collection of paths we obtain the bound $O(n + m)$ as a special case.

Finally, combining Lemma 2.3 with the data structures that answer level ancestor queries in constant time, we get the following corollary.

Corollary 2.1. *There are data structures, requiring $O(n)$ preprocessing, which resolve the following query in $O(\log n)$ time: Given a node v_i, a point x on the edge $(v_i, v_{p(i)})$, and a real $r \le d(x, v_1)$, find a node v_j, an ancestor of v_i, and a point y on the edge $(v_j, v_{p(j)})$ such that $d(x, y) = r$.*

We will also need the next result which follows directly from the above discussion.

Lemma 2.4. *There are data structures, requiring $O(n)$ preprocessing, which resolve the following query in constant time: Given a pair of paths, $P[v_i, v_j]$ and $P[v_k, v_t]$, determine whether the pair intersects, and find the two end points (nodes) of the path $P[v_i, v_j] \cap P[v_k, v_t]$ when it is nonempty.*

Proof. Suppose without loss of generality that $LCA(v_i, v_j) = v_i$ and $LCA(v_k, v_t) = v_k$. (Otherwise, consider the intersections between the four relevant pairs among the set $\{P[v_i, LCA(v_i, v_j)], \quad P[v_j, LCA(v_i, v_j)], P[v_k, LCA(v_k, v_t)], P[v_t, LCA(v_k, v_t)]\}$.) If $LCA(v_i, v_k) \notin \{v_i, v_k\}$, then the paths do not intersect. Suppose without loss of generality that $LCA(v_i, v_k) = v_i$. Note that $v_k \in P[v_i, v_j]$ if and only if $LCA(v_j, v_k) = v_k$. If $v_k \notin P[v_i, v_j]$ the paths do not intersect. Otherwise, the intersection of the two paths is the path $P[v_k, LCA(v_j, v_t)]$. □

3. Unweighted 1-center problem with addends of a collection of subtrees

Given is a finite collection of subtrees $\mathcal{T} = \{T_i\}_{i\in I}$ of a tree T and their associated addends. Notice that we can assume without loss of generality that the addends are nonnegative since we are solving minimax problems. If $T_i \bigcap T_j = \emptyset$ let $P[T_i, T_j]$ be the unique simple path connecting T_i and T_j. The unweighted 1-center problem with addends of $\mathcal{T}$ is to find $x^* \in T$ satisfying

$$\max_{i\in I}\{d(x^*, T_i) + k_i\} = \min_{x\in A(T)} \max_{i\in I}\{d(x, T_i) + k_i\}.$$

x^* is called an *unweighted 1-center with addends of $\mathcal{T}$*.

The optimal value is at least $\max_{q\in I} k_q$. For each pair of subtrees T_i, T_j, the optimal solution value to the unweighted 1-center problem with addends for the subcollection $\{T_i, T_j\}$ is clearly $\max\{(d(T_i, T_j)+k_i+k_j)/2, k_i, k_j\}$. Due to the Helly property, (see Kolen and Tamir (1990) [24]), the optimal solution value to the unweighted 1-center problem with addends for $\mathcal{T}$ is

$$\max\left\{\max_{i,j\in I}(d(T_i, T_j) + k_i + k_j)/2; \max_{i\in I}\{k_i\}\right\}.$$

Thus, a necessary and sufficient condition to have an optimal value greater than $k^* = \max_{i\in I} k_i$ is that there exists a pair of subtrees T_i, T_j such that $d(T_i, T_j) + k_i + k_j > 2k^*$. We next show how to test efficiently whether the optimal solution value is equal to k^*. For each $i \in I$, define $T_i' = \{x \in A(T) : d(x, T_i) \le k^* - k_i\}$. Then the optimal value is equal to k^* if and only if $\cap_{i\in I}\{T_i'\}$ is nonempty. For $i \in I$, let z_i be the closest point to v_1 in T_i'. Let $t \in I$ satisfy

$$d(z_t, v_1) = \max_{i\in I}\{d(z_i, v_1)\}.$$

By Lemma 2.1, (replace T_i by T_i' and x_i by z_i), $\cap_{i\in I}\{T_i'\}$ is nonempty, if and only if $z_t \in \cap_{i\in I}\{T_i'\}$. Thus, it is sufficient to compute the distances $d(z_i, v_1)$, $i \in I$, identify z_t explicitly on $A(T)$, and check whether z_t is contained in T_i' for all $i \in I$.

As above, we let x_i to be the closest point to v_1 in T_i. If $d(x_i, v_1) \leq k^* - k_i$, then $d(z_i, v_1) = 0$. Otherwise, $d(z_i, v_1) = d(x_i, v_1) + k_i - k^*$. Thus, by the results in Section 2, in $O(n + m + \log n)$ time we find the point z_t. Finally, we note that for each $i \in I$, $z_t \in T_i'$ if and only if $d(z_t, T_i) \leq k^* - k_i$. Again, using the results in Section 2, we conclude that the total effort to determine whether the optimal solution value to the unweighted 1-center problem with addends is equal to k^* is $O(n + \sum_{i\in I} n_i^l)$. In this case, z_t, defined above, is an optimal solution.

Without loss of generality suppose that $T_1, T_2 \in \mathcal{T}$ satisfy

$$d(T_1, T_2) + k_1 + k_2 = \max_{i,j\in I}\{d(T_i, T_j) + k_i + k_j\}.$$

For the ease of readability, we denote the following inequality by (A1):

(A1): $d(T_1, T_2) + k_1 + k_2 > 2\max_{q\in I} k_q$.

Lemma 3.1. *Assume that* (A1) *holds. Then x^*, the point of $P[T_1, T_2]$ such that $d(x^*, T_1) = (d(T_1, T_2) - k_1 + k_2)/2$, is an unweighted 1-center with addends of $\mathcal{T}$. In particular, the optimal value is $(d(T_1, T_2) + k_1 + k_2)/2$.*

Proof. When (A1) is satisfied we have $d(T_1, T_2) + k_1 + k_2 > 2\max\{k_1, k_2\}$. Thus, x^* is clearly the unweighted 1-center with addends of $\{T_1, T_2\}$. In particular, $d(T_1, T_2) > 0$. It is sufficient to show that for each $i \in I$, $d(T_i, x^*) + k_i \leq d(T_1, x^*) + k_1 = d(T_2, x^*) + k_2 = (d(T_1, T_2) + k_1 + k_2)/2$. If $x^* \in T_i$, then from (A1) $d(T_i, x^*) + k_i = k_i < (d(T_1, T_2) + k_1 + k_2)/2$. Suppose that $x^* \notin T_i$, and let y_i be the closest point to T_i on $P[T_1, T_2]$. (If T_i intersects $P[T_1, T_2]$, define y_i to be the closest point to x^* in T_i.) Without loss of generality suppose that $y_i \in P[x^*, T_2]$. If $d(T_i, x^*) + k_i > d(T_2, x^*) + k_2$, we obtain the contradiction

$$\begin{aligned} d(T_1, T_i) + k_1 + k_i &= d(T_1, x^*) + k_1 + d(T_i, x^*) + k_i > d(T_1, x^*) + k_1 + d(T_2, x^*) + k_2 \\ &= d(T_1, T_2) + k_1 + k_2. \quad \square \end{aligned}$$

Lemma 3.2. *Assume that* (A1) *holds. Let T' be a subtree in $A(T)$, not necessarily in $\mathcal{T}$, which does not intersect the interior of $P[T_1, T_2]$. Then* $\max_{i\in I}\{d(T', T_i) + k_i\} = \max\{d(T', T_1) + k_1, d(T', T_2) + k_2\}$.

Proof. Note that property (A1) implies that $d(T_1, T_2) > 0$. Suppose, $T_p \in \mathcal{T}$ is such that:

$$d(T', T_p) + k_p = \max_{i\in I}\{d(T', T_i) + k_i\}.$$

Case 1. First we analyze the case when T_p intersects $P[T_1, T_2]$. Assume by contradiction that $d(T', T_p) + k_p > \max\{d(T_1, T') + k_1, d(T_2, T') + k_2\}$. Let y be the closest point to T' on $P[T_1, T_2]$. Suppose first that $y \notin T_p$, and assume without loss of generality that T_p intersects $P[T_1, y]$. Then, $d(T', T_p) + k_p > d(T', T_1) + k_1$ would imply that $d(y, T_p) + k_p > d(y, T_1) + k_1$. Hence,

$$\begin{aligned} d(T_2, T_p) + k_2 + k_p &= d(T_2, y) + k_2 + d(y, T_p) + k_p > d(T_2, y) + k_2 + d(y, T_1) + k_1 \\ &= d(T_1, T_2) + k_1 + k_2. \end{aligned}$$

The latter contradicts the maximality of the pair T_1, T_2. Suppose now that $y \in T_p$. Assume first that y is in the interior of $P[T_1, T_2]$. Then,

$$d(T', y) + k_p \geq d(T', T_p) + k_p > d(T', T_1) + k_1 = d(T', y) + d(y, T_1) + k_1.$$

Hence, $k_p > d(y, T_1) + k_1$. Similarly,

$$d(T', y) + k_p \geq d(T', T_p) + k_p > d(T', T_2) + k_2 = d(T', y) + d(y, T_2) + k_2,$$

and $k_p > d(y, T_2) + k_2$. Thus, we obtain the following contradiction,

$$2k_p > d(T_1, y) + k_1 + d(y, T_2) + k_2 = d(T_1, T_2) + k_1 + k_2.$$

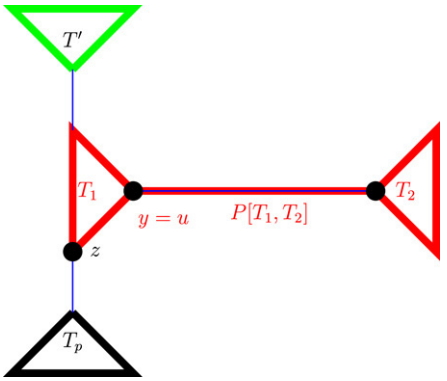

Fig. 1. Illustration of the proof of Lemma 3.2. Case 2-a when $y = u$.

If y is not an interior point, assume without loss of generality that $y \in T_1$. In this case, $d(T_1, T_p) = 0$, and therefore, the inequality $k_1 + k_p \leq d(T_1, T_2) + k_1 + k_2 = d(y, T_2) + k_1 + k_2$, implies $k_p \leq d(y, T_2) + k_2$. Hence,

$$d(T', T_p) + k_p \leq d(T', y) + k_p \leq d(T', y) + d(y, T_2) + k_2 = d(T', T_2) + k_2.$$

Case 2. Next we consider the case when T_p does not intersect $P[T_1, T_2]$. Let $\bar{T}$ be the minimal subtree including $T_1 \bigcup T_2 \bigcup T'$. Let z be a closest point to T_p in $\bar{T}$. If z is not unique, i.e., when T_p intersects $\bar{T}$, set z to be the closest point to $P[T_1, T_2]$ in $T_p \cap \bar{T}$. Let y be the closest point on $P[T_1, T_2]$ to T'. Suppose first that y is not in the interior of $P[T_1, T_2]$. Without loss of generality assume that $y \in T_1$. We distinguish two cases: $z \in P[T_1, T_2]$ and $z \notin P[T_1, T_2]$. (a) $z \notin P[T_1, T_2]$. (See Fig. 1.) Let u be the closest point to T_p on $P[T_1, T_2]$. If $y \neq u$, using the maximality property of T_1 and T_2, i.e., $d(T_p, T_1) + k_p + k_1 \leq d(T_1, T_2) + k_1 + k_2$, we have $d(T_p, u) + k_p \leq d(u, T_2) + k_2$. Hence, $d(T', T_p) + k_p \leq d(T', T_2) + k_2$. Suppose that $y = u$. In this case, from the maximality of T_1, T_2, we have $d(T_p, T_2) + k_p + k_2 \leq d(T_1, T_2) + k_1 + k_2$. The latter implies that $d(T_p, u) + k_p \leq d(T_1, u) + k_1 \leq d(T_2, u) + k_2$. (Note that the second inequality follows directly from condition (A1).) Therefore,

$$d(T', T_p) + k_p \leq d(T', u) + d(u, T_p) + k_p \leq d(T', u) + d(u, T_2) + k_2 = d(T', T_2) + k_2.$$

(b1) $z \in P[T_1, T_2]$, and $z \neq y$. (See Fig. 2.) In this case we must have $d(z, T_p) + k_p \leq d(z, T_2) + k_2$, since otherwise, $d(T_1, T_p) + k_1 + k_p > d(T_1, T_2) + k_1 + k_2$, contradicting the maximality of the pair T_1, T_2. Therefore,

$$d(T', T_2) + k_2 \leq d(T', T_p) + k_p \leq d(T', z) + d(z, T_p) + k_p \leq d(T', z) + d(z, T_2) + k_2 = d(T', T_2) + k_2.$$

(b2) $z \in P[T_1, T_2]$, and $z = y$. (See Fig. 3.) In this case, from the maximality of T_1, T_2, we have $d(T_p, T_2) + k_p + k_2 \leq d(T_1, T_2) + k_1 + k_2$. The latter implies that $d(T_p, z) + k_p \leq d(T_1, z) + k_1 \leq d(T_2, z) + k_2$. Therefore,

$$d(T', T_p) + k_p \leq d(T', z) + d(z, T_p) + k_p \leq d(T', z) + d(z, T_2) + k_2 = d(T', T_2) + k_2.$$

Next, suppose that $y \in P[T_1, T_2]$, $y \notin T_1$, $y \notin T_2$ and $d(y, T_1) + k_1 \leq d(y, T_2) + k_2$. Let $\alpha = (d(T_1, T_2) - k_1 + k_2)/2$, $\alpha_1 = d(y, x^*)$ and $\alpha_2 = d(y, T_1)$. (Recall that x^* is the 1-center with addends of the collection $\mathcal{T}$.) Notice that from construction $\alpha = \alpha_1 + \alpha_2$, $\alpha_1 \geq 0$ and $\alpha_2 > 0$. Let $\beta = d(T', y)$. Note that $\beta > 0$. Let $\gamma = d(z, T_p) + k_p$. To conclude the proof we perform a subcase analysis distinguishing the following subcases:

(a′) $z \in T'$

$$d(T', T_p) + k_p \leq d(z, T_p) + k_p = \gamma < \gamma + \beta \leq \alpha_2 + k_1.$$

Notice that the last inequality holds since otherwise $d(T_2, T_p) + k_2 + k_p > d(T_1, T_2) + k_1 + k_2$. Therefore

$$d(T', T_2) + k_2 \leq d(T', T_p) + k_p < \alpha_2 + k_1 \leq \alpha + k_1 \leq d(y, T_2) + k_2 \leq d(T', T_2) + k_2.$$

(b′) $z \in P[y, T']$, $z \neq y$. Let $\beta = \beta_1 + \beta_2$, where $\beta_1 = d(z, y) > 0$ and $\beta_2 = d(z, T')$. Again, from the maximality of the pair T_1, T_2, it is clear that:

$$\gamma < \beta_1 + \gamma \leq \alpha_2 + k_1.$$

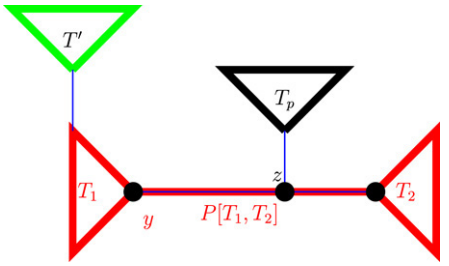

Fig. 2. Illustration of the proof of Lemma 3.2. Case 2-(b1).

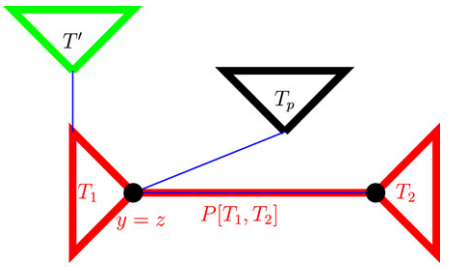

Fig. 3. Illustration of the proof of Lemma 3.2. Case 2-(b2).

Thus,

$$d(T_1, T') + k_1 = \beta_2 + \beta_1 + \alpha_2 + k_1 > \beta_2 + \gamma = d(T', T_p) + k_p.$$

(c′) $z \in T_2$ and $z \notin P[T_1, T_2]$. In this case since $d(T_1, T_p)+k_1+k_p \leq d(T_1, T_2)+k_1+k_2$ then $d(z, P[T_1, T_2])+k_p \leq k_2$. Hence,

$$d(T_2, T') + k_2 \geq d(T_p, T') + k_p.$$

(d′) $z \in T_1$, $z \notin P[T_1, T_2]$. Following the argument in case (c′), we obtain

$$d(T_1, T') + k_1 \geq d(T_p, T') + k_p.$$

(e′) $z \in P[T_1, T_2]$. Assume first that $z \in P[y, T_2]$. Then, from the maximality of T_1 and T_2, $d(z, T_p) + k_p \leq d(z, T_2) + k_2$. Therefore, $d(T', T_2) + k_2 \leq d(T', T_p) + k_p \leq d(T', T_2) + k_2$. Next assume that $z \in P[y, T_1]$. In this case, $d(z, T_1) + k_1 \geq d(z, T_p) + k_p$, and hence $d(T', T_1) + k_1 \geq d(T', T_p) + k_p \geq d(T', T_1) + k_1$. □

Lemma 3.3. *Assume that* (A1) *holds. Let T' be a subtree of $A(T)$, not necessarily in $\mathcal{T}$, which intersects the interior of $P[T_1, T_2]$, but does not intersect some subtree T_i, $i \in I$. Let $T_p \in \mathcal{T}$ be a subtree satisfying*

$$d(T', T_p) + k_p = \max_{i \in I}\{d(T', T_i) + k_i\}.$$

Suppose that $d(T_p, T') + k_p > \max_{i \in I} k_i$. Then either:

$$\max_{i \in I} k_i < d(T', T_p) + k_p = \max\{d(T', T_1) + k_1, d(T', T_2) + k_2\},$$

or T_p does not intersect the interior of $P[T_1, T_2]$.

Proof. Assume that T_p intersects the interior of $P[T_1, T_2]$. Suppose without loss of generality that,

$$T_p \cap P[T_1, T_2] \subset P[T', T_2].$$

Note that T_p does not intersect T' since we assume that $d(T_p, T')+k_p > \max_{i \in I} k_i$. If $d(T', T_p)+k_p > d(T', T_2)+k_2$ then

$$d(T_1, T_p) + k_p + k_1 > d(T_1, T_2) + k_1 + k_2$$

which contradicts the maximality property of T_1 and T_2. Hence,

$$d(T', T_2) + k_2 \geq d(T', T_p) + k_p. \quad \square$$

3.1. Algorithmic implications

The above lemmas imply the validity of the following algorithm to solve the unweighted 1-center problem with addends for a collection $\mathcal{T}$. By the results in Section 2 we first find a point which is not in $\cap_{i\in I}\{T_i\}$, in $O(n+\sum_{i\in I} n_i^l)$ time, or conclude that the optimal value is $\max_{i\in I} k_i$.

Algorithm 3.1.

1. Find a point $x \in A(T)$ which does not intersect all subtrees in $\mathcal{T}$ or conclude that the optimal value is $\max_{i\in I} k_i$.
2. Let T_p be such that $d(x, T_p) + k_p = \max_{i\in I}\{d(x, T_i) + k_i\}$.
3. Let T_r be such that $d(T_p, T_r) + k_r = \max_{i\in I}\{d(T_p, T_i) + k_i\}$.
4. Let T_s be such that $d(T_r, T_s) + k_s = \max_{i\in I}\{d(T_r, T_i) + k_i\}$. Stop. x^*, the point of $P[T_r, T_s]$ such that $d(x^*, T_r) = (d(T_r, T_s) - k_r + k_s)/2$, is an unweighted 1-center with addends of $\mathcal{T}$. In particular, the optimal value is $(d(T_r, T_s) + k_r + k_s)/2$.

Applying the results in Section 2 we conclude with the next results.

Theorem 3.1. *Let $\mathcal{T} = \{T_i\}$, $i \in I$, be a collection of m subtrees. For $i \in I$, let n_i^l denote the number of leaves of T_i plus 1. The unweighted 1-center problem with addends of $\mathcal{T}$ can be solved in $O(n + \sum_{i\in I} n_i^l)$ time.*

Theorem 3.2. *Let $\mathcal{T} = \{T_i\}$, $i \in I$, be a collection of m paths. The unweighted 1-center problem with addends of $\mathcal{T}$ can be solved in $O(n + m)$ time.*

Theorem 3.3. *Let $\mathcal{T} = \{T_i\}$, $i \in I$, be a collection of m neighborhood subtrees. The unweighted 1-center problem with addends of $\mathcal{T}$ can be solved in $O(n + m)$ time.*

3.2. Unweighted 1-center problem with no addends

In this subsection we specialize to the case where there are no addends associated with the subtrees in $\mathcal{T}$, i.e., $k_i = 0$ for all $i \in I$. The above lemmas imply the validity of the following algorithm to solve this specialized model for a collection $\mathcal{T}$ which does not intersect at a point of $A(T)$.

Algorithm 3.2.

1. Find $T_q \in \mathcal{T}$ which does not intersect all subtrees in $\mathcal{T}$.
2. Let T_p be such that $d(T_q, T_p) = \max_{i\in I}\{d(T_q, T_i)\}$.
3. Let T_r be such that $d(T_p, T_r) = \max_{i\in I}\{d(T_p, T_i)\}$.
4. Let T_s be such that $d(T_r, T_s) = \max_{i\in I}\{d(T_r, T_i)\}$. Stop, $d(T_r, T_s) = \max_{i,j\in I}\{d(T_i, T_j)\}$.

We next show the redundancy of Step 4 if we initiate Step 1 with a neighborhood subtree. For general subtrees, Step 4 is essential, as illustrated by the following example.

Example 3.1. Define $T = (V, E)$ by $V = \{v_1, v_2, v_3, v_4\}$, and $E = \{(v_1, v_3), (v_2, v_3), (v_3, v_4)\}$. Let $d(v_1, v_3) = 3$, $d(v_2, v_3) = 2$, and $d(v_3, v_4) = 1$. Finally, $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$, where $T_1 = \{v_1\}$, $T_2 = \{v_2\}$, $T_3 = P[v_1, v_2]$ and $T_4 = \{v_4\}$. If we apply the above algorithm initiating with $T_q = T_3$, we obtain $T_p = T_4$, $T_r = T_1$, and $T_s = T_2$.

Lemma 3.4. *Assume that $\cap\{T_i\}_{i\in I}$ is empty. Let T' be a neighborhood subtree in $A(T)$, not necessarily in $\mathcal{T}$, which does not intersect some subtree T_j, $j \in I$. Let $T_p \in \mathcal{T}$, satisfy $d(T', T_p) = \max_{i\in I}\{d(T', T_i)\}$. Then $d(T', T_p) = \max\{d(T', T_1), d(T', T_2)\}$.*

Proof. Using Lemma 3.2, we assume without loss of generality that T' intersects the interior of $P[T_1, T_2]$. We can assume that T_p does not intersect the path $P[T_1, T_2]$, since otherwise

$$\max\{d(T', T_1), d(T', T_2)\} \leq d(T', T_p) \leq \max\{d(T', T_1), d(T', T_2)\}.$$

Let $z \in A(T)$ be the center of the neighborhood T', and let $r \geq 0$ be its radius. Since $z \notin T_1 \cap T_2$, assume without loss of generality that $z \notin T_1$. Let $x \in P[T_1, T_2]$ be the closest point to z in $P[T_1, T_2]$. Note that $x \in T'$, since T' intersects $P[T_1, T_2]$. Let $\bar{T}$ be the minimal subtree of $A(T)$ containing T_1, T_2 and z. From the maximality of the path $P[T_1, T_2]$, we can assume that T_p does not intersect $\bar{T}$, and that the closest point to T_p in $\bar{T}$, say y is in $P[T_1, z] \cup P[x, T_2] = P[T_1, T_2] \cup P[x, z]$. Suppose first that $y \in P[T_1, z]$. From the maximality of $P[T_1, T_2]$, we have $d(x, T_p) \leq d(x, T_1)$. Hence,

$$0 < d(T_p, T') = d(T_p, z) - r \leq d(T_p, x) + d(x, z) - r \leq d(T_1, x) + d(x, z) - r = d(T_1, z) - r \leq d(T_1, T').$$

Next, suppose that $y \in P[x, T_2]$. From the maximality of $P[T_1, T_2]$, we have $d(x, T_p) \leq d(x, T_2)$. Hence,

$$\begin{aligned} 0 < d(T_p, T') &= d(T_p, z) - r = d(T_p, x) + d(x, z) - r \leq d(T_2, x) + d(x, z) - r \\ &= d(T_2, z) - r \leq d(T_2, T'). \quad \square \end{aligned}$$

The last lemma suggests that we can start the algorithm with an arbitrary point (degenerate subtree T'). This choice simplifies the above algorithm.

Algorithm 3.3.

1. Let x be an arbitrary point of $A(T)$. If $x \in T_i$, for each $i \in I$: Stop, all subtrees have a common point.
2. Let T_p be such that $d(x, T_p) = \max_{i \in I}\{d(x, T_i)\} > 0$. Let $y \in T_p$ satisfy $d(x, y) = d(x, T_p)$. If $y \in T_i$, for each $i \in I$: Stop, all subtrees have a common point.
3. Let T_r be such that $d(T_p, T_r) = \max_{i \in I}\{d(T_p, T_i)\}$. Stop, $d(T_r, T_p) = \max_{i,j \in I}\{d(T_i, T_j)\}$.

Remark 3.1. Note that if x is not a common point of all subtrees in the collection $\mathcal{T}$, then $\mathcal{T}$ has a common point if and only if the point y is a common point.

The above algorithm extends the classical algorithm of Handler [18] from 1973, which applies to the case where each subtree in the collection is a point in $A(T)$.

4. Unweighted 2-center problem with addends of a collection of subtrees

Given the collection $\mathcal{T} = \{T_i\}_{i \in I}$ of subtrees and their associated addends, the unweighted 2-center problem with addends is to find a pair of points $X^* = \{x_1^*, x_2^*\}$ such that:

$$\max_{i \in I}\{d(X^*, T_i) + k_i\} = \min_{X:|X|=2} \max_{i \in I}\{d(X, T_i) + k_i\}.$$

First of all, we can assume without loss of generality that (A1) holds. Otherwise the 2-center problem reduces to a 1-center problem and the optimal value is $\max_{i \in I} k_i$. In this case the optimal solution can be found in linear time solving one 1-center problem with addends.

Lemma 4.1. *Suppose that* (A1) *holds and let T_i, T_j be a diametrical pair of I. Let x be an interior point of $P[T_i, T_j]$. Let $A_i(T)$, $A_j(T)$, be a diametrical partition with respect to T_i, T_j and x, i.e., $A_i(T)$, $A_j(T)$ are subtrees of $A(T)$,*

$$A_i(T) \cup A_j(T) = A(T), \qquad A_i(T) \cap A_j(T) = \{x\}, \qquad T_i \subset A_i(T), \qquad T_j \subset A_j(T).$$

Define $\mathcal{T}_i = \{T_t^i = T_t \cap A_i(T) : t \in I\}$, and $I' = \{t \in I : T_t^i \neq \emptyset\}$. If $\max_{t,u \in I'}\{d(T_t^i, T_u^i) + k_t + k_u\} > \max_{t \in I'} 2k_t$, then there exists a subtree $T_s^i \in \mathcal{T}_i$ such that T_i, T_s^i is a diametrical pair of $\mathcal{T}_i$.

Proof. First we note that for each pair of (nonempty) subtrees T_t^i and T_u^i in $\mathcal{T}_i$, $d(T_t^i, T_u^i) = d(T_t, T_u)$. Suppose that the subtrees T_q^i and T_l^i, where $T_q^i \neq T_i$ and $T_l^i \neq T_i$, are diametrical for the collection $\mathcal{T}_i$. From the supposition in the lemma it follows that $d(T_q^i, T_l^i) > 0$, and therefore the path $P[T_q^i, T_l^i]$ is well defined. Let z be the closest point of T_i to x. (Note that $z \neq x$.) To prove the result we distinguish the following cases. First, assume that $z \in P[T_q, x]$. Then,

in this case $T_q^i = T_q$. Since T_i, T_j are diametrical for $\mathcal{T}$ we have $k_i \geq d(T_q, z) + k_q$. Suppose that $z \in P[T_l, x]$. Then, by the same argument, $T_l^i = T_l$ and $k_i \geq d(T_l, z) + k_l$. Thus,

$$d(T_q, T_l) + k_q + k_l \leq d(T_q, z) + d(z, T_l) + k_q + k_l \leq 2k_i = d(T_i, T_i) + 2k_i.$$

The above implies that the pair T_i, T_i is also a diametrical pair of $\mathcal{T}_i$. Suppose now that $z \notin P[T_l, x]$. If $z \in T_l$, then we have

$$d(T_q, T_l) + k_q + k_l \leq d(T_q, z) + k_q + k_l \leq k_i + k_l = d(T_i, T_l) + k_i + k_l.$$

The above implies that the pair T_i, T_l^i is also a diametrical pair of $\mathcal{T}_i$. If $z \notin T_l$, we clearly have $z \in P[T_l, T_q]$. Hence, using the above inequality, $k_i \geq d(T_q, z) + k_q$, we obtain

$$d(T_l, T_q) + k_l + k_q = d(T_l, z) + d(z, T_q) + k_l + k_q \leq d(T_l, z) + k_l + k_i = d(T_l, T_i) + k_l + k_i.$$

The above implies that the pair T_i and T_l^i is diametrical for $\mathcal{T}_i$. This concludes the proof for the case $z \in P[T_q, x]$. A symmetric proof validates the case $z \in P[T_l, x]$. Thus, it is sufficient to consider the case where $z \notin P[T_q, x] \cup P[T_l, x]$. In particular, $z \notin P[T_q, T_l]$. Next, consider the case where $P[T_q, T_l]$ does not intersect $P[T_i, x]$, and let y be the closest point to $P[T_q, T_l]$ on $P[T_i, x]$. At least one of the pair $\{T_q, T_l\}$ does not intersect $P[T_i, x]$. Suppose that $T_l \cap P[T_i, x] = \emptyset$. Then $y \in P[T_i, T_l]$. Suppose that $y \in T_q$. Then

$$d(T_i, T_l) = d(T_i, y) + d(y, T_l) \geq d(T_i, y) + d(T_q, T_l). \tag{1}$$

Using the maximality of (T_q, T_l), in $\mathcal{T}_i$ and (1) we have

$$k_l + k_q + d(T_q, T_l) \geq k_l + k_i + d(T_l, T_i) \geq k_l + k_i + d(T_i, y) + d(T_q, T_l).$$

Hence,

$$k_q \geq k_i + d(T_i, y). \tag{2}$$

The maximality of (T_i, T_j) in $\mathcal{T}$ implies

$$k_i + d(T_i, y) \geq k_l + d(T_l, y). \tag{3}$$

Combining (2) and (3) we obtain $k_q \geq k_l + d(T_l, y)$. Therefore,

$$2k_q \geq k_q + k_l + d(T_l, y) \geq k_q + k_l + d(T_l, T_q).$$

The above contradicts the condition in the lemma which requires $k_q + k_l + d(T_q, T_l) > 2 \max_{t \in I'} k_t$.

If $y \notin T_q$, then, from the fact that $\{T_i, T_j\}$ is a diametrical pair,

$$d(T_q, T_l) + k_q + k_l \leq d(T_l, y) + d(y, T_q) + k_q + k_l \leq d(T_l, y) + d(y, T_i) + k_l + k_i = d(T_l, T_i) + k_l + k_i.$$

The result holds with $T_s^i = T_l^i$. Next, consider the case where $P[T_q, T_l]$ intersects $P[T_i, x]$. (See Fig. 4.) Let x_l (x_q) be the closest point to T_l (T_q) in $P[T_q, T_l] \cap P[T_i, x]$. Without loss of generality suppose that $x_l \in P[x_q, x]$. (If $x_q = x_l$ and this point is an endpoint of $P[T_q, T_l]$, we assume, for convenience, that this is the endpoint of T_l.) Then, $x_l \in P[T_i, T_l]$. From the fact that $\{T_i, T_j\}$ is a diametrical pair,

$$d(T_l, T_q) + k_q + k_l = d(T_l, x_l) + d(x_l, T_q) + k_q + k_l \leq d(T_l, x_l) + d(x_l, T_i) + k_l + k_i = d(T_l, T_i) + k_l + k_i.$$

The result holds with $T_s^i = T_l^i$. □

Remark 4.1. We note that the above result holds even if we replace the collection $\mathcal{T}_i$ by a subcollection induced by any subset $I'' \subseteq I'$, such that $i \in I''$.

Remark 4.2. The reader can check that in the 2-center problems without addends if $A_i(T)$, $A_j(T)$ is a diametrical partition with respect to T_i, T_j and x (x being interior to $P[T_i, T_j]$), then there always exists a subtree $T_s^i \in \mathcal{T}_i$ such that T_i, T_s^i is a diametrical pair of $\mathcal{T}_i$. Nevertheless, the additional condition in the above lemma is essential when addends are present, as illustrated in the next example.

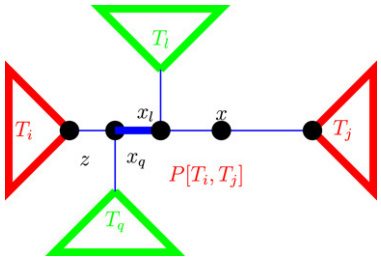

Fig. 4. Illustration of the proof of Lemma 4.1 when $P[T_q, T_l] \cap P[T_i, x] \neq \emptyset$.

Example 4.1. Consider the path network with node set $V = \{v_1, v_2, v_3\}$ and edge set $E = \{(v_1, v_2), (v_2, v_3)\}$. The edge lengths are: $d(v_1, v_2) = 1$, $d(v_2, v_3) = 2$. The family of subtrees and their corresponding addends are: $T_1 = \{v_1\}$, $k_1 = 0$; $T_2 = \{(v_1, v_2), (v_2, v_3)\}$, $k_2 = 1.25$; and $T_3 = \{v_3\}$ and $k_3 = 0$. The unique diametrical pair for the entire collection is T_1, T_3. If we establish a diametrical partition with respect to x^*, the 1-center solution, which is the midpoint of $P[T_1, T_3]$, then $\mathcal{T}_1 = \{T_1, T_2^1\}$, and $I' = \{1, 2\}$. In the subtree induced by these two elements T_1 is not an element in the diametrical pair of I'. The unique diametrical pair of I' is T_2^1, T_2^1.

Theorem 4.1. *Let x^* be a solution to the 1-center problem with addends on $\mathcal{T}$, and let T_i, T_j be a pair of diametrical subtrees. Suppose that $d(T_i, T_j) + k_i + k_j > 2 \max_{l \in I} k_l$. Let $A_i(T), A_j(T)$ be a diametrical partition of $A(T)$ with respect to T_i, T_j and x^*. Define*

$$\mathcal{T}_i = \{T_l : l \in I, d(T_l, T_i) + k_i \leq d(T_l, T_j) + k_j\}, \quad \mathcal{T}_j = \mathcal{T} \setminus \mathcal{T}_i.$$

Let x_i^* (x_j^*) be the solution to the 1-center problem with addends for $\mathcal{T}_i$ ($\mathcal{T}_j$) on $A_i(T)$ ($A_j(T)$). Then $\{x_i^*, x_j^*\}$ is a solution to the unweighted 2-center problem with addends for $\mathcal{T}$ on $A(T)$.

Proof. Let $\{y_1, y_2\}$ be an optimal solution to the unweighted 2-center problem with addends for $\mathcal{T}$, and let r_2^* be the optimal objective value. Consider the pair of points, $\{x_i^*, x_j^*\}$ as a feasible solution to the 2-center problem with addends for $\mathcal{T}$. Suppose without loss of generality that its objective value is determined by x_i^*, the solution to the 1-center problem with addends for $\mathcal{T}_i$. Hence, by the above lemma, the 2-center objective value corresponding to $\{x_i^*, x_j^*\}$ is either $k_r = \max_{T_t^i \in \mathcal{T}_i} k_t$, or there exists a subtree $T_s^i \in \mathcal{T}_i$ such that the pair T_i, T_s^i is a diametrical pair for $\mathcal{T}_i$, and the 2-center objective value corresponding to $\{x_i^*, x_j^*\}$ is $(d(T_i, T_s) + k_i + k_s)/2$. If the value of the 2-center problem corresponding to $\{x_i^*, x_j^*\}$ is k_r, then $r_2^* \geq \max_{t \in I}\{k_t\} \geq k_r$. The latter implies that $\{x_i^*, x_j^*\}$ is also an optimal solution. Hence, suppose that the value of the 2-center problem corresponding to $\{x_i^*, x_j^*\}$ is $(d(T_i, T_s) + k_i + k_s)/2$. Suppose without loss of generality that $d(T_i, y_1) \leq d(T_i, y_2)$, i.e., T_i is "served" by y_1. If T_s is also served by y_1, then $r_2^* \geq \max(d(T_i, y_1) + k_i; d(T_s, y_1) + k_s) \geq (d(T_i, T_s) + k_i + k_s)/2$. The latter then implies that $\{x_i^*, x_j^*\}$ is also an optimal solution. Hence, suppose that T_s is served by y_2, i.e., $d(T_s, y_2) < d(T_s, y_1)$. Consider first the case where T_s does not contain x^*, the solution of the 1-center with addends. If y_1 serves T_j then $r_2^* \geq \max(d(T_i, y_1) + k_i; d(T_j, y_1) + k_j) \geq (d(T_i, T_j) + k_i + k_j)/2 \geq (d(T_i, T_s) + k_i + k_s)/2$. The latter then implies that $\{x_i^*, x_j^*\}$ is also an optimal solution. Hence, suppose that T_j is served by y_2. In this case

$$r_2^* \geq \frac{d(T_s, T_j) + k_s + k_j}{2} = \frac{d(T_s, x^*) + k_s}{2} + \frac{d(x^*, T_j) + k_j}{2} = \frac{d(T_s, x^*) + k_s}{2} + \frac{d(x^*, T_i) + k_i}{2}$$
$$\geq \frac{d(T_i, T_s) + k_i + k_s}{2}.$$

Again, it follows that $\{x_i^*, x_j^*\}$ is also an optimal solution. Suppose now that T_s contains x^*. Since $T_s^i \in \mathcal{T}_i$ we have $d(T_i, T_s) + k_i \leq d(T_j, T_s) + k_j$. If y_1 serves T_j then, as above, $r_2^* \geq \max(d(T_i, y_1) + k_i; d(T_j, y_1) + k_j) \geq (d(T_i, T_j) + k_i + k_j)/2 \geq (d(T_i, T_s) + k_i + k_s)/2$, and $\{x_1^*, x_2^*\}$ is also an optimal solution. Finally, suppose that T_j is served by y_2. In this case, since $d(T_i, T_s) + k_i \leq d(T_j, T_s) + k_j$, we have

$$r_2^* \geq (d(T_s, T_j) + k_s + k_j)/2 \geq (d(T_i, T_s) + k_i + k_s)/2.$$

Again, it follows that $\{x_i^*, x_j^*\}$ is also an optimal solution. □

The above theorem generalizes the classical result of Handler [19], who proved that in order to solve the unweighted 2-center problem without addends for nodes on a tree, one has to split the tree at the middle point of its diameter, and solve the respective two unweighted 1-center problems. The next theorem summarizes the algorithmic implication of the above result.

Theorem 4.2. *The unweighted* 2*-center problem with addends of a collection of subtrees* $\mathcal{T}$ *can be solved by solving three* 1*-center problems.*

5. Weighted 1-center problem with addends of a collection of subtrees

Given is the collection of subtrees $\mathcal{T} = \{T_i\}_{i\in I}$, and a set of reals $\{k_i\}_{i\in I}$. For each $i \in I$, define

$$f_i(x) = w_i[d(x, T_i) + k_i].$$

Consider the function $f(x)$, defined on $A(T)$, and the following optimization problem:

$$f(x) = \max_{i\in I} f_i(x),$$

$$\min_{x\in A(T)} f(x). \tag{4}$$

This problem can also be rewritten as:

$$\begin{aligned} \min_{x\in A(T)} \quad & z \\ & z \geq w_i[d(T_i, x) + k_i], \quad i \in I. \end{aligned}$$

We note that the above problem is a special case of a much more general, non-convex, single facility location problem on a tree, discussed in Tamir and Halman (2005) [35]. As a result it can be solved by the general algorithm in [35] in $O((n + (\sum_{i\in I} n_i^l)) \log n)$ time, (see Theorem 4.3 in [35]). We next show a simple specialized $O(n + (\sum_{i\in I} n_i^l) \log n)$ algorithm.

Algorithm 5.1.

1. Find a centroid $\bar{v}$ of T', the current subtree search domain. (Initially, the subtree search domain is the original tree T.) The complexity of this step is $O(n')$, where n' is the number of nodes in T'. (See Kariv and Hakimi (1979) [22].)
2. Compute $f(\bar{v})$, the objective value at $\bar{v}$. The complexity of this step is $O(n' + \sum_{i\in I} n_i^l)$, if we apply the results in Section 2.
3. Identify $I'' = \{i \in I : f(\bar{v}) = f_i(\bar{v})\}$. The complexity of this step is again $O(n' + \sum_{i\in I} n_i^l)$.
   (a) If there exists $i \in I''$ such that $\bar{v} \in T_i$, stop. $\bar{v}$ is optimal. (For any $x \in A(T)$, $f(x) \geq f_i(x) \geq w_i k_i = f_i(\bar{v}) = f(\bar{v})$.)
   (b) If there exists a pair of distinct indices $i, j \in I''$, such that T_i and T_j are in distinct connected components of $F(T' : \bar{v})$, stop. $\bar{v}$ is optimal.
   (c) If there exists a connected component T^q of $F(T' : \bar{v})$ such that $T_i \subseteq T^q$, for all $i \in I''$, then there is an optimal solution x^* which is contained either in T^q or on the unique edge connecting $\bar{v}$ to T^q. Define T' to be the subtree induced by T^q and $\bar{v}$ as the current search domain.
   (d) If T' has more than two nodes, repeat. If T' is an edge of T, say $T' = P[v_r, v_s]$, for some pair of adjacent nodes in V, x^*, the solution to the 1-center problem is defined by the solution to the following two-dimensional linear programming problem:
   $$\begin{aligned} \min_{x\in P[v_r,v_s]} \quad & z \\ & z \geq w_i[d(T_i, x) + k_i], \quad i \in I. \end{aligned}$$
   The above linear program can be solved in $O(|I|)$ time by the algorithm in Megiddo (1983) [28].

The complexity of the above algorithm is clearly $O(n + (\sum_{i\in I} n_i^l)\log n)$, since we use centroid decomposition, and therefore there are $O(\log n)$ iterations. When each subtree T_i is a path then $n_i^l \leq 3$ and therefore the complexity reduces to $O(n + m\log n)$. Thus, we have

Theorem 5.1. *The weighted* 1*-center problem with addends of a collection of subtrees* $\mathcal{T} = \{T_i : i \in I\}$ *can be solved in* $O(n + (\sum_{i\in I} n_i^l)\log n)$ *time. If for each* $i \in I$, T_i *is a path the time reduces to* $O(n + m\log n)$.

Remark 5.1. The weighted 1-center problem with addends of a collection of neighborhoods can be solved in $O(m + n)$ time by a simple modification of the linear time algorithm in Megiddo (1983) [28] to solve the weighted 1-center problem of a collection of points. In addition, the weighted 1-center problem with addends of a collection of subtrees on a path graph can be solved in $O(m)$ since it reduces to a linear program with $4m$ constraints and 2 variables. In fact, the problem can be solved in linear time even when the nodes (points on the line) are not sorted. Also, note that in the case of a path, subtrees are actually subpaths.

6. Weighted and unweighted p-center problems with addends of a collection of subtrees $\{T_i\}_{i\in I}$

Given a set of real addends, $\{k_i\}$, $i \in I$ and nonnegative weights $\{w_i\}_{i\in I}$; the weighted p-center problem with addends is to find a set of p points, $X_p = \{x_1, \ldots, x_p\}$, $X_p \subseteq A(T)$, minimizing

$$\max_{i\in I}\{w_i(d(T_i, X_p) + k_i)\}.$$

We have assumed that each subtree T_i is discrete. Hence, in this case we may assume that $p < n$, and $p < m$. The above model is a special case of the round-trip weighted p-center problem studied in Tamir and Halman (2005) [35]. To see the connection, for each T_i let z_i be the closest point of T_i to v_1, the root of T, and let L_i^* denote the set of leaves of T_i. Now T_i can be represented as the union of the $|L_i^*|$ paths connecting z_i to the leaves of T_i. In particular, $d(T_i, X_p) = \min_{y\in L_i^*} d(P[z_i, y], X_p)$. Next, for each center $x_j \in X_p$, and leaf $y \in L_i^*$, $d(P[z_i, y], x_j)$, the distance of x_j from the path $P[z_i, y]$, satisfies the linear equation $2d(P[z_i, y], x_j) + 2d(z_i, y) = rt(x_j, z_i, y, x_j)$, where $rt(x_j, z_i, y, x_j)$ is the length of the tour on T, starting at x_j, visiting z_i and y and returning to x_j. (See Section 4.2 in [35].) Thus, $w_i(d(P[z_i, y], x_j)+k_i) = 2w_i rt(x_j, z_i, y, x_j)+w_i(k_i-2d(z_i, y))$. In Tamir and Halman z_i represents a customer and its respective set L_i^* represents the set of depots where z_i can be served. As a special case, our model can be solved in $O(mn\log(m+n))$ by the algorithms in Tamir and Halman (2005) [35]. These algorithms are based on the general parametric approach of Megiddo (1979, 1983) [26,27], and they have the same uniform complexity for all values of p. In particular, even for the case where the number of centers p is fixed, and each of the subtrees is a path, we obtain superquadratic algorithms. This is certainly a shortcoming, since in most applications, p is significantly smaller than n and m. Thus, there is a need for algorithms which are more efficient for these cases. This is what we focus on. To illustrate, the algorithm we develop here has $O((n + m\log n)(\log n + \log m))$ complexity when applied to the case when p is fixed and each subtree is a path.

6.1. Solving the weighted covering problem with addends

Given a set of reals $\{r_i\}_{i\in I}$, the *minimum covering problem* is to find a set X of minimal cardinality such that:

$$d(T_i, X) + k_i \leq r_i/w_i, i \in I.$$

For a prescribed integer p, the *p-covering decision problem* is to determine whether the solution to the minimum covering problem is at most p. (Note that in this subsection the reals $\{k_i\}$ and $\{r_i\}$ are not assumed to be nonnegative. Also, to simplify the notation we rename the values r_i/w_i, $\forall i \in I$, as r_i.) Without loss of generality we assume that $r_i \geq k_i$ for all $i \in I$, since otherwise there is no solution to the problem. We set $r_i' = r_i - k_i$, for $i \in I$, and reformulate the problem.

Find a subset $X \subseteq A(T)$ of minimum cardinality, such that

$$d(T_i, X) \leq r_i',\ i \in I.$$

For $i \in I$, define $T_i' = \{x \in A(T) : d(x, T_i) \leq r_i'\}$. Then, the covering problem is equivalent to finding $X \subseteq A(T)$ of minimum cardinality, such that $T_i' \cap X$ is nonempty for each $i \in I$. To solve the covering problem we use the

following algorithm based on Theorem 6.5 in Kolen and Tamir (1990) [24]. It is the simplification of the respective covering algorithm in Tamir and Halman (2005) [35].

Algorithm 6.1.

1. Suppose that the tree is rooted at v_1. For each T_i', $i \in I$, let z_i, be the closest point in T_i' to v_1.
2. Let $Z = \{z_i : i \in I\}$. If v_1 is not already in Z, augment it to Z. Partially order $\{z_i\}$ by the partial order induced by the rooted tree, i.e. $z_i \preceq z_j$ iff $z_j \in P[z_i, v_1]$.
3. Select a minimal point with respect to the partial order, $z_k \in Z$. Add z_k to X (initially $X = \emptyset$).
4. Let $I_k = \{i \in I : z_k \in T_i'\}$, $I := I \setminus I_k$, $Z := Z \setminus \{z_i : i \in I_k\}$. If $I = \emptyset$, stop; X is a solution to the minimum covering problem. Otherwise go to step 3.

Remark 6.1. When we apply the above algorithm to solve the p-covering decision problem we can stop also when the cardinality of the covering set X exceeds the prescribed value p. In particular, Steps 3–4 are iterated at most $\min(n, m, p+1) = p+1$ times.

6.1.1. Complexity analysis

- *Step* 1. We assume that each T_i is discrete. For each $i \in I$, if we root T at x_i, the root of T_i, and scan the tree T from top to leaves it takes $O(n)$ time to find $V_i' \subseteq V$, the set of all the nodes of T_i', and the set $Y_i \subseteq A(T)$, consisting of all the leaves of T_i', which are not nodes. Note that Y_i has at most one point on each edge of T. Hence, $|Y_i| \leq n-1$. (Actually, if T has n' leaves, $|Y_i| \leq n'$.) For convenience, each point in Y_i is recorded by the edge it belongs to, and its distances from the two nodes of that edge. For each T_i', let $v_{q(i)}$ be the closest node to the tree root v_1 in V_i'. We also assume that for each node $v_t \in V$ we have a list, L_t of all T_i', $i \in I$, containing v_t. Compute the nodes $\{v_{q(i)}\}_{i\in I}$ as follows: Starting at v_1, scan T from root to leaves. Reaching a node v_t do:
  – consider the list L_t of trees T_i'.
  – for each $i \in I$ such that $T_i' \in L_t$, set $v_{q(i)} = v_t$ and remove i from I.
  – if I is empty stop.

  Next generate the points $\{z_i\}_{i\in I}$ as follows: If there is a leaf of T_i' on the edge connecting $v_{q(i)}$ to its father in V, set z_i to be equal to that leaf. Otherwise, set $z_i = v_{q(i)}$. The complexity of the above step is clearly $O(mn)$.
- *Step* 2. From top to bottom generate in $O(m+n)$ time the partial ordering on $\{z_i\}_{i\in I}$ with the exception that the points $\{z_i\}$ on any given edge are not ordered. (We want to avoid the effort of ordering the points $\{z_i\}$ that are on the same respective edge, since that will require $O(n + m \log m)$ additional time.)
- *Step* 3. The cardinality of Z is at most m. To identify a minimal point $z_k \in Z$ with respect to the partial order, we consider some minimal edge, say $(v_s, v_w) \in E$ containing points in Z. If v_s is the father of v_w, then the minimal point of Z on this edge is the one closest to v_w. We define this point as z_k.
- *Steps* 4. To identify the set of subtrees $\{T_i'\}$ containing z_k, consider the lists L_s and L_w, consisting of all subtrees containing the node v_s and v_w, respectively. If $T_i' \in L_w$, then from the minimality of z_k, $z_k \in T_i'$. Hence, suppose that $T_i' \in L_s \setminus L_w$. If $z_k = v_s$, then clearly $z_k \in T_i'$. If $z_k \neq v_s$, then $z_k \in T_i'$ if and only if T_i' has a leaf in Y_i, say y' such that $d(y', v_s) \geq d(z_k, v_s)$.

Each iteration of Steps 3 and 4 takes $O(m)$ time, using the preprocessing done in Step 2. Therefore, the complexity of the above algorithm is $O(mn)$.

Remark 6.2. The $O(nm)$ complexity of the above implementation is determined by the preprocessing and is independent of p. Hence, we suggest a different implementation which avoids the above preprocessing. As we will shortly see, it is particularly useful for solving the decision problem, where Steps 3–4 are iterated at most p times.

We implement the above algorithm differently to obtain improved bounds for cases where $\sum_{i\in I} n_i^l = o(mn)$.

Algorithm 6.2.

1. Suppose that the tree is rooted at v_1. For each T_i', $i \in I$, let z_i be the closest point in T_i' to v_1.
2. Let $Z = \{z_i : i \in I\}$. If v_1 is not already in Z, augment it to Z. Compute $d(z_i, v_1)$ for all $i \in I$. (We do not explicitly generate the set Z.)

3. Let $z_k \in Z$ satisfy $d(z_k, v_1) = \max\{d(z_i, v_1) : z_i \in Z\}$. Find explicitly the location of z_k in $A(T)$, i.e., find the edge of $P[x_k, v_1]$ containing z_k, and add it to X (initially $X = \emptyset$).
4. Let $I_k = \{i \in I : z_k \in T_i'\}$, $I := I \setminus I_k$, $Z := Z \setminus \{z_i : i \in I_k\}$. If $I = \emptyset$, stop; X is a solution to the minimum covering problem. Otherwise go to step 3.

6.1.2. Complexity analysis

The complexity of Step 2 is $O(m)$. ($d(z_i, v_1) = \max\{0, d(x_i, v_1) - r_i'\}$). The effort to find z_k explicitly in Step 3 is $O(m + \log n)$ by the results in Section 2. Finally, the effort to execute Step 4 is $O(\sum_{i \in I} n_i^l)$, by the results in Section 2, (see also Section 3), since it is dominated by the time needed to compute $d(z_k, T_i)$ for all $i \in I$. Therefore, with this implementation, the total time needed to determine whether the minimum covering size is at most p is $O(n + p \log n + p \sum_{i \in I} n_i^l)$.

6.1.3. Special cases

For a collection of paths when $T_i = P[a_i, b_i]$ for each $i \in I$, the time needed for resolving the decision problem is only $O(n + p \log n + pm)$. (Kolen (1985) [23] describes an $O(mn)$ algorithm for this case.)

When each subtree T_i, $i \in I$, is a neighborhood, centered at some point in $A(T)$, so is T_i'. Hence, the covering problem can be solved in $O(n + m)$ time by the algorithms in Slater (1976) [34] and Kariv and Hakimi (1979) [22]. When the tree T is a path then we can solve the covering decision problem scanning the path, and since we assume that there are n nodes the complexity is $O(n)$.

6.2. Solving the weighted p-center problem with addends of a collection of discrete subtrees $\{T_i\}_{i \in I}$

As noted above, the problem can be solved in $O(mn \log(m + n))$ time by applying the algorithms in Tamir and Halman (2005) [35]. These algorithms are based on the parametric approach of Megiddo (1979, 1983) [26,27]. We first show a direct search procedure, which is easier to implement, but has a slightly worst complexity, i.e., $O(mn \log^2(m + n))$. Then, in the following sections we present the algorithms which are significantly more efficient when the number of centers p is "smaller" than m and n, e.g., when p is fixed. We start by identifying a set containing the optimal value r_p^* for the weighted case with addends. We claim that r_p^* is either equal to $\max_{i \in I}\{w_i k_i\}$, or is an element in the set

$$R = \{(d(T_i, T_j) + k_i + k_j)/(1/w_i + 1/w_j) : i, j \in I\}.$$

To verify the above, it is sufficient to consider the single center case. Hence, assume that x is an optimal solution to the 1-center problem. Then if the optimal solution value r_1^*, is greater than $\max_{i \in I}\{w_i k_i\}$, there is a pair of subtrees, T_i and T_j such that

$$w_i(d(x, T_i) + k_i) = w_j(d(x, T_j) + k_j) = r_1^*.$$

Therefore, there is a pair of nodes, $v_s \in T_i$, and $v_t \in T_j$, such that $x \in P[v_s, v_t]$ and $w_i(d(v_s, x) + k_i) = w_j(d(v_t, x) + k_j) = r_1^*$. The latter implies that $r_1^* = (d(v_s, v_t) + k_i + k_j)/(1/w_i + 1/w_j)$.

In the unweighted case without addends r_p^*, the optimal objective value, is an element in the set $\{d(v_t, v_s)/2 : v_s, v_t \in V\}$. As in the classical unweighted p-center problem for nodes, (see Frederickson and Johnson (1983) [12]), the effort to find r_p^* is dominated by the time needed to solve $O(\log n)$ covering problems. Hence, the total complexity is $O(mn \log n)$. In the particular case that T is a path since the decision problem is solved in $O(n)$ the p-center can be solved in $O(n \log n)$. Next we discuss the weighted case with or without addends, and assume without loss of generality that r_p^* is an element in the set $R = \{(d(T_i, T_j) + k_i + k_j)/(1/w_i + 1/w_j) : i, j \in I\}$, which in turn is a subset of $\{(d(v_s, v_t) + k_i + k_j)/(1/w_i + 1/w_j) : v_s, v_t \in V, i, j \in I\}$. We can refine the definition of a superset containing R as follows: For each subtree T_i and a node $v_s \in T_i$, augment a node v_s^i to the tree T, and connect it to v_s with an edge of length k_i. The augmented tree, $T' = (V', E')$, will have $O(nm)$ nodes. It is then clear that R is contained in the set

$$R' = \{d(v_s^i, v_t^j)/(1/w_i + 1/w_j) : v_s^i, v_t^j \in V', i, j \in I\}.$$

With the above representation of R', we can search for r_p^* in this set, using the method in Megiddo and Tamir (1983) [29] with the improvement by Cole (1987) [7]. (To determine whether a given $r \in R'$ satisfies $r_p^* \leq r$ or

not, we apply the covering problem algorithm from the previous section.) The augmented tree T' has $O(mn)$ nodes, and therefore with these search procedures, we can identify r_p^* in $O(mn\log^2(mn))$ time. In the particular case that T is a path the p-center problem reduces to $O(n\log^2(mn))$ since the covering problem is solved in $O(n)$ time.

Applying the above results for the weighted model with nonnegative addends to the unweighted case with nonnegative addends, we conclude that in the latter case the problem has an optimal solution value r_p^* equal to

$$\max_{i\in I} k_i,$$

or equal to an element in the set

$$R'' = \{(d(v_s, v_t) + k_i + k_j)/2 : i, j \in I; v_s, v_t \in V\}.$$

As above we define a superset containing R''. Specifically, for each subtree T_i and a node $v_s \in T_i$, augment a node v_s^i to the tree T, and connect it to v_s with an edge of length k_i. The augmented tree, $T' = (V', E')$, will have $O(nm)$ nodes. It is then clear that R'' is contained in the set

$$\overline{R''} = \{d(v_s^i, v_t^j)/2 : v_s^i, v_t^j \in V', i, j \in I\}.$$

We can now apply the procedure in Frederickson and Johnson (1983) [12], using the covering problem to determine whether a given element $r \in \overline{R''}$, satisfies $r_p^* \le r$ or not. Since T' has $O(mn)$ nodes, the total time to find r_p^* is $O(mn\log(mn))$. We summarize the above results in the following theorem.

Theorem 6.1. *The weighted p-center problem with nonnegative addends of a collection of subtrees $\mathcal{T} = \{T_i : i \in I\}$ can be solved in $O(mn\log^2(mn))$ time. In the unweighted case without addends the time reduces to $O(mn\log n)$ while in the case with addends the time is $O(mn\log(mn))$.*

The above solution approach does not provide improved complexity bounds to solve the p-center problem for fixed $p \ge 2$. The bounds stated in the last theorem are independent of p, and the best bound that we have even for the 2-center problem is still $O(mn\log^2(mn))$. We next propose a different algorithm which is based on the general parametric approach in Megiddo [26,27]. This new algorithm is more efficient when p is "small" relative to n, and $\sum_{i\in I} n_i^l = o(nm)$. The algorithm is based on parametrization of the above $O(n + p\log n + p\sum_{i\in I} n_i^l)$ algorithm to resolve the covering decision problem.

6.3. The parametric algorithm I

We apply the parametric approach of Megiddo (1983) [28], to solve the p-center problem, using the above Algorithm 6.2 which resolves the decision problem, as a master algorithm. Consider the following parametric decision problem: Is there a set X, with $|X| \le p$, such that

$$d(T_i, X) \le r/w_i - k_i, \quad i \in I.$$

r_p^*, the optimal value of the weighted p-center problem with addends, is clearly the smallest value of the parameter r, for which the answer to the decision problem is affirmative. The general parametric approach of Megiddo, suggests that in our application, we simulate the above algorithm for the decision problem parametrically, without specifying a value for the parameter r. Specifically, in Step 2, we can use m processors working in $O(1)$ parallel time, (phases). Each processor will be assigned a point x_i and compute the respective distance $d(z_i(r), v_1)$ in $O(1)$ time. Thus, the total effort to find $d(z_i(r), v_1)$, for all $i \in I$, is dominated by the time we need to apply the decision problem algorithm $O(\log m)$ times plus $O(m)$ extra time. In Step 3 we need to compute $\max_{i\in I}\{d(z_i(r), v_1)\}$. We use a parallel sorting algorithm (as in Cole (1987) [7]). The total effort to find $d(z_k(r), v_1)$ is dominated by the time we need to apply the decision problem algorithm $O(\log m)$ times plus $O(m\log m)$ extra time. In this step we also need to find the edge of T containing $z_k(r)$, i.e., explicitly locate $z_k(r)$ in $A(T)$. By Lemma 2.3 this can be done in $O(\log n)$ time, by a single processor. Thus, the total effort to find the edge containing $z_k(r_p^*)$, will require $O(\log n)$ applications of the decision problem algorithm plus $O(\log n)$ extra time. In Step 4 we need to compute the distance $d(z_k(r), T_i)$ for each $i \in I$, and compare it with $r/w_i - k_i$. This can be done in constant time in parallel, by $O(\sum_{i\in I} n_i^l)$ processors. Each processor will compute the distance of $z_k(r)$ from a path connecting a root of some subtree T_i to one of the leaves of T_i. Thus, the total effort per each iteration of Step 4 in the parametric implementation is dominated by the time

required to solve $O(\log \sum_{i\in I} n_i^l)$ decision problems plus $O(\sum_{i\in I} n_i^l)$ extra time. We iterate Steps 3–4 $O(p)$ times. To conclude the total time to solve the weighted p-center problem with nonnegative addends with the parametric approach is

$$O\left(\left(p\left(\log\left(\sum_{i\in I} n_i^l\right)\right)+p\log n\right)\left(n+p\log n+p\sum_{i\in I} n_i^l\right)\right).$$

6.4. The parametric algorithm II

We note that for the case where $\log m \le p$, we can obtain the slightly better bound

$$O\left(\left(p\left(\log\left(\sum_{i\in I} n_i^l\right)\right)+\log m\log n\right)\left(n+p\log n+p\sum_{i\in I} n_i^l\right)\right),$$

by applying the parametric approach to the the following version of the above covering algorithm.

Algorithm 6.3.

1. Suppose that the tree is rooted at v_1. For each T_i', $i \in I$, let z_i be the closest point in T_i' to v_1.
2. Let $Z = \{z_i : i \in I\}$. If v_1 is not already in Z, augment it to Z. Compute $d(z_i, v_1)$ for all $i \in I$. Sort the set $\{d(z_i, v_1) : i \in I\}$. Explicitly generate the set Z, by finding for each z_i, $i \in I$, the respective edge of $P[x_i, v_1]$, containing z_i.
3. Let $z_k \in Z$ satisfy $d(z_k, v_1) = \max\{d(z_i, v_1) : z_i \in Z\}$. Add z_k to X (initially $X = \emptyset$).
4. Let $I_k = \{i \in I : z_k \in T_i'\}$, $I := I \setminus I_k$, $Z := Z \setminus \{z_i : i \in I_k\}$. If $I = \emptyset$, stop; X is a solution to the minimum covering problem. Otherwise go to step 3.

In the parametrization of the above algorithm, to explicitly find the points $\{z_i(r)\}$, $i \in I$, in Step 1 we can use m processors working in $O(\log n)$ parallel time, (phases). Each processor will be assigned a point x_i and locate the respective point $z_i(r)$ in $O(\log n)$ time, (based on Lemma 2.3). Thus, the total effort to find for all points $z_i(r)$, $i \in I$, the respective edges containing $z_i(r_p^*)$, will require $O(\log n \log m)$ applications of the decision problem algorithm plus $O(m \log n)$ extra time. Next we find the partial ordering (corresponding to r_p^*), of all points $z_i(r)$, (including the ordering within each edge). We use a parallel sorting algorithm (as in Cole (1987) [7]). The total effort for this step is dominated by the time we need to apply the decision problem algorithm $O(\log m)$ times plus $O(m \log m)$ extra time. Finally, we have to iterate Steps 3–4 $O(p)$ times. At a given iteration we need to compute the distances from some point $z_k(r)$ to the remaining subtrees $\{T_i\}$. This can be done in constant time in parallel, by $O(\sum_{i\in I} n_i^l)$ processors. Each processor will compute the distance of $z_k(r)$ from a path connecting a root of some subtree T_i to one of the leaves of T_i. Thus, the total effort per each iteration of Step 4 in the parametric implementation is dominated by the time required to solve $O(\log \sum_{i\in I} n_i^l)$ decision problems plus $O(\sum_{i\in I} n_i^l)$ extra time. To conclude the total time to solve the weighted p-center problem with nonnegative addends with the second parametric approach is

$$O\left(\left(p\left(\log\left(\sum_{i\in I} n_i^l\right)\right)+\log m\log n\right)\left(n+p\log n+p\sum_{i\in I} n_i^l\right)\right).$$

Theorem 6.2. *The weighted p-center problem with addends of a collection of subtrees $\mathcal{T} = \{T_i : i \in I\}$ can be solved in*

$$O\left(\left(p\left(\log\left(\sum_{i\in I} n_i^l\right)\right)+\min(p, \log m)\log n\right)\left(n+p\log n+p\sum_{i\in I} n_i^l\right)\right).$$

6.4.1. Special cases

A collection of paths. In this case $n_i^l \leq 3$ and the complexity for solving the p-center problem reduces to

$$O\left((p\log m + \min(p, \log m)\log n)\,(n + p\log n + pm)\right).$$

In particular, the 2-center model for a collection of paths is solvable in

$$O((n+m)(\log n + \log m))$$

time. For comparison purposes, the 1-center case is solved in Section 5 in $O(n + m\log n)$ time.

A collection of neighborhoods. Suppose that for each $i \in I$, T_i is a neighborhood centered at the point $y_i \in I$, with radius r_i, i.e.,

$$T_i = \{x \in A(T) : d(x, y_i) \leq r_i\}.$$

From the above discussion we conclude that in this case, the optimal objective value, r_p^* is either equal to $\max_{i\in I}\{k_i\}$, or r_p^* is an element in the set $R = \{(d(T_i, T_j) + k_i + k_j)/(1/w_i + 1/w_j) : i, j \in I\}$, where $d(T_i, T_j) = d(y_i, y_j) - r_i - r_j$. Thus

$$R = \{(d(y_i, y_j) - r_i - r_j + k_i + k_j)/(1/w_i + 1/w_j) : i, j \in I\}.$$

We can assume without loss of generality that the points $\{y_i : i \in I\}$ are nodes of T. Hence, T has $O(n + m)$ nodes. With the above representation of R, we can search for r_p^* in this set, using the method in Megiddo and Tamir (1983) [29] with the improvement by Cole (1987) [7]. (To determine whether a given $r \in R'$ satisfies $r_p^* \leq r$ or not, we apply the above $O(n + m)$ covering problem algorithm from the previous section.) The augmented tree T' has $O(m + n)$ nodes, and therefore with these search procedures, we can identify r_p^* in $O((m + n)\log^2(m + n))$ time.

7. Final comments and open problems

In the center problems considered above the p servers can be located anywhere in $A(T)$. This version is usually labelled as a continuous model. In the discrete version servers are restricted to a finite set, e.g., the node set of the tree. We note in passing that the algorithms presented in Sections 3, 5 and 6 can easily be modified to the discrete case, without increasing the complexity. The results in Section 4 about the continuous unweighted 2-center model are not known to be extendable to the discrete case. (The latter statement applies even to the discrete, unweighted 2-center problem for the case where the customers are represented by the nodes of the tree.) A generalization of the above discrete center problem is the following covering problem with setup costs for the serving facilities (servers). In this model each subtree customer T_i, $i \in I$, is associated with a service radius r_i. The setup cost of establishing a facility at node v_j, $j = 1, \ldots, n$, is $c_j \geq 0$. The goal is to establish facilities with minimum total setup cost, such that for each T_i there is a facility within a distance of r_i from T_i. This problem is polynomially solvable for neighborhood subtrees, and NP-hard even for covering paths on a star tree with $r_i = 0$ for all $i \in I$, (Kolen and Tamir (1990) [24]). Finally we pose a few open problems.

1. We have presented above an $O(n + m\log n)$ algorithm to solve the weighted 1-center problem for a collection of m paths. Can we obtain a better complexity bound e.g., $O(n + m)$, by reducing the size of I by a constant factor at each iteration?
2. Given a set of radii $\{r_i'\}$, $i \in I$, we have presented in Section 6 an $O(n + p\log n + pm)$ algorithm to determine whether there is a p-covering for a collection of m paths. The question is whether there are subquadratic algorithms in terms of p and m to solve this problem. The papers by Gavril [13–15] might be useful in this regard. We note that the special case where $r_i' = 0$ for all $i \in I$, can be solved in $O(n + m)$ time by a simple implementation of Algorithm 6.2.
3. We have described in Section 3 a linear time algorithm to solve the unweighted 1-center problem with addends for a general collection of subtrees. When each subtree is a point our algorithm reduces to Handler's classical algorithm. Hence, it is an interesting and insightful question to determine whether the 1-center problem for general subtrees or even paths can actually be transformed or reduced to one with points. (It is easy to verify that such a transformation does exist for a collection of neighborhood subtrees.) In view of the above mentioned NP-hardness result it seems

unlikely that for a general integer p, the p-center problem for a collection of paths can be transformed to an equivalent model with a collection of points.

Acknowledgments

This work started during a visit of the second author to the University of Seville. It was partially supported by research grants of the Spanish Ministry of Science and Education BFM2002-10418, MTM2006-15054, MTM2004-0909 and the Andalusian Consejeria de Innovación Ciencia y Empresa FQM-331 and P06-FQM-01366.

References

[1] M.A. Bender, M. Farach-Colton, The level ancestor problem simplified, in: Proceedings of the 5-th Latin American Symposium on Theoretical Informatics, LATIN'02, in: LNCS, vol. 2286, Springer-Verlag, 2002, pp. 508–515.
[2] O. Berkman, U. Vishkin, Finding level- ancestors in trees, Journal of Computer and System Sciences 48 (1994) 214–230.
[3] O. Berman, D. Simchi-Levi, A. Tamir, The minimax multistop location problem, Networks 18 (1988) 39–43.
[4] R.W. Bulterman, F.W. van der Sommen, G. Zwaan, T. Verhoeff, A.J.M. van Gasteren, W.H.J. Feijen, On computing a longest path in a tree, Information Processing Letters 81 (2002) 93–96.
[5] R. Chandrasekaran, A. Daughety, Location on tree networks: p-center and N-dispersion problems, Mathematics of Operations Research 6 (1981) 50–57.
[6] A. Chan, R.L. Francis, A round-trip location problem on a tree graph, Transportation Science 10 (1976) 35–51.
[7] R. Cole, Slowing down sorting networks to obtain faster algorithms, Journal of the ACM 34 (1987) 168–177.
[8] J.M. Díaz-Báñez, J.A. Mesa, A. Schöbel, Continuous location of dimensional structures, European Journal of Operational Research 152 (2004) 22–44.
[9] P.F. Dietz, Finding level-ancestors in dynamic trees, in: Proceedings of Workshop on Algorithms and Data Structures, WADS'91, 1991, pp. 32–40.
[10] G.N. Frederickson, Optimal algorithms for partitioning trees and locating p-centers in trees, Technical Report, Department of Computer Science, Purdue University West Lafayette, IN, 1990.
[11] G.N. Frederickson, Optimal algorithms for tree partitioning, in: Proceedings 2nd Symposium on Discrete Algorithms, SODA'91, San-Francisco, 1991, pp. 186–177.
[12] G.N. Frederickson, D.B. Johnson, Finding k-th paths and p-centers by generating and searching good data structures, Journal of Algorithms 4 (1983) 61–80.
[13] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques and maximum independent set of a chordal graph, SIAM Journal on Computing 1 (1972) 180–187.
[14] F. Gavril, A recognition algorithm for intersection graphs of directed paths in directed trees, Discrete Mathematics 13 (1975) 237–249.
[15] F. Gavril, A recognition algorithm for intersection graphs of paths in trees, Discrete Mathematics 23 (1978) 211–227.
[16] S.L. Hakimi, Optimum location of switching centers and the absolute centers and medians of a graph, Operations Research 12 (1964) 450–459.
[17] S. Halfin, On finding the absolute and vertex centers of a tree with distances, Transportation Science 8 (1974) 75–77.
[18] G.Y. Handler, Minimax location of a facility in an undirected tree graph, Transportation Science 7 (1973) 287–293.
[19] G.Y. Handler, Finding two centers of a tree: The continuous case, Transportation Science 12 (1978) 93–106.
[20] D. Harel, R.E. Tarjan, Fast algorithms for finding nearest common ancestors, SIAM Journal on Computing 13 (1984) 338–355.
[21] T.C. Huang, J.-C. Lin, H.-J. Chen, A self-stabilizing algorithm which finds a 2-center of a tree, Computers and Mathematics with Applications 40 (2000) 607–624.
[22] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems: Part 1. The p-centers, SIAM Journal on Applied Mathematics 37 (1979) 539–560.
[23] A. Kolen, The round-trip p-center and covering problem on a tree, Transportation Science 19 (1985) 222–234.
[24] A. Kolen, A. Tamir, Covering problems, in: P.B. Mirchandani, R.L. Francis (Eds.), Discrete Location Theory, Wiley, 1990 (Chapter 6).
[25] C.C. Lin, On vertex addends in minimax location problems, Transportation Science 9 (1975) 165–168.
[26] N. Megiddo, Combinatorial optimization with rational objective functions, Mathematics of Operations Research 4 (1979) 414–424.
[27] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, Journal of the ACM 30 (1983) 852–865.
[28] N. Megiddo, Linear time algorithms for linear programming in R^3 and related problems, SIAM Journal on Computing 12 (1983) 759–776.
[29] N. Megiddo, A. Tamir, New results on the complexity of p-center problems, SIAM Journal on Computing 12 (1983) 751–758.
[30] N. Megiddo, A. Tamir, E. Zemel, R. Chandrasekaran, 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 (1981) 328–337.
[31] S. Nickel, J. Puerto, A.M. RodríguezChía, An approach to location models involving sets as existing facilities, Mathematics of Operations Research 28 (2003) 693–715.
[32] F. Plastria, Continuous covering location problems, in: Z. Drezner, H.W. Hamacher (Eds.), Facility Location: Applications and Theory, Springer, Berlin, Heidelberg, New York, 2002 (Chapter 2).
[33] J. Puerto, A. Tamir, Locating tree-shaped facilities using the ordered median objective, Mathematical Programming 102 (2005) 313–338.
[34] P.S. Slater, R-domination in graphs, Journal of the ACM 23 (1976) 446–450.
[35] A. Tamir, N. Halman, One-way and round-trip center location problems, Discrete Optimization 2 (2005) 168–184.
[36] A. Tamir, E. Zemel, Locating centers on a tree with discontinuous supply and demand regions, Mathematics of Operations Research 7 (1982) 183–197.