Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Algorithms 56 (2005) 50–75
Journal of Algorithms
www.elsevier.com/locate/jalgor

Conditional location of path and tree shaped facilities on trees

A. Tamir [a,*], J. Puerto [b], J.A. Mesa [c], A.M. Rodríguez-Chía [d]

[a] *School of Mathematical Sciences, Tel Aviv University, Israel*
[b] *Facultad de Matemáticas, Universidad de Sevilla, Spain*
[c] *Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Spain*
[d] *Facultad de Ciencias, Universidad de Cádiz, Spain*

Received 22 September 2001

Available online 24 March 2005

Abstract

In this paper we deal with the location of extensive facilities on trees, both discrete and continuous, under the condition that existing facilities are already located. We require that the selected new server is a subtree, although we also specialize to the case of paths. We study the problem with the two most widely used criteria in Location Analysis: center and median. Our main results under the center criterion are nestedness properties of the solution and subquadratic algorithms for the location of paths and subtrees. For the case of the median criterion we prove that unlike the case where there is no existing facility, the continuous conditional median subtree problem is NP-hard and we develop a corresponding fully polynomial approximation algorithm. We also present subquadratic algorithms for almost all other models.

1. Introduction

In a typical location problem there is a set of demand points embedded in some metric space and the objective is to locate a specified number of servers optimizing some

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

0196-6774/$ – see front matter
doi:10.1016/j.jalgor.2005.01.005

criterion, which usually depends on the distances between the demand points and their respective servers. The criteria mostly considered are the minimization of the average service distance or the maximum distance. These criteria are referred to as the median and the center problems respectively. In many practical situations we already have some servers located in the underlying space and they provide service to the customers. These servers can not be relocated. However, sometimes an increased budget is available for establishing more servers (facilities) to improve service, e.g., decrease service distances. Minieka [18] has coined the term *conditional location problem* for situations where there already exists a set of servers. (Models with these features already appeared in [21].) Minieka solved the conditional median and center location problem on networks in the presence of only one additional center (one point). Several papers dealing with conditional location models have appeared since then. Most deal with the situation where there is only one existing facility present. References discussing point-conditional location problems on networks are [4,7, 8,12]. There are also studies in the literature dealing with the point-conditional location problem in the plane. (See for example, [5,6].)

The papers cited above focus on location problems where a server (facility) is represented by a point in the metric space. However, in recent years there has been a growing interest in studying the location of connected structures, which cannot be represented by (isolated) points in the space. These studies were motivated by concrete decision problems related to routing and network design. For instance, in order to improve the mobility of the population and reduce traffic congestion, many existing rapid transit networks are being updated by extending or adding lines. These lines can be viewed as new facilities, and the issue of deciding the place of the alignment and the location of stations on a new line, can be categorized as a conditional extensive facility location problem. Moreover, since the cost of constructing one unit of length and that of the stations can be estimated, a budget constraint can be expressed in terms of the total length of the new facility (line). Other potential applications appear in hierarchical network design such as the case where a high power transmission or a cable communication network must be extended.

The first studies on location of connected structures (which we call *extensive* facilities) appeared in the early eighties [3,13,19,20,22,28]. Hakimi et al. [11] focused on the complexity of solving many versions of location problems of extensive facilities. The different versions are derived by considering such elements as locating one or several facilities, whether the facilities are paths or tree shaped, whether the underlying network is a tree or a general graph, and the objective function used. Researchers have followed the suggested classification, and improved the results for many of the models listed in [11]. For example we cite, [1,15,23,24,30,32]. More relevant references are mentioned throughout the paper.

Almost all papers which focus on extensive facilities do not consider the case where servers (facilities) may already exist. We are aware of only two references which take into consideration existing facilities. The solution to the problem of locating a path minimizing the median function in the presence of an existing point-facility not belonging to the path was applied by Becker and Perl [3] to design an algorithm for the 2-core of a tree. (A 2-core of a tree is a pair of paths minimizing the sum of the weighted distances from the nodes of the graph to their respective closest path.) They solve their conditional model in $O(n)$ time, and find the 2-core in $O(n^2)$ time. An improved linear time algorithm for a 2-core of a tree has been recently presented by Wang [33]. An $O(n \log n)$ time algorithm was also

obtained by Mesa [17] for the conditional path center problem in the pure topological case (unweighted and equal edge lengths) of a tree. As mentioned above, conditional location problems of extensive facilities pertain to a realistic class of location problems where a given extensive facility is already located and one looks for the installation of a new extensive facility without altering the position and shape of the existing one. We require the new facility to be connected but the two facilities may be disconnected.

In this paper we study conditional location problems on trees where the new facility is required to be connected. Topologically, the selected server has to be a subtree. We also specialize to the important case where the connected structure is further restricted to be a path of the network. We consider discrete and continuous versions of both the center and the median objectives. Constraints are expressed in terms of total length of the extensive facility. Comparing our results with those known for the unconditional versions, we note that our algorithms are more complex. Polynomial complexity is preserved for all models but one. The continuous unconditional median subtree problem is linearly solvable [29], while we prove that the respective conditional version is NP-hard. We provide a fully polynomial time approximation scheme for this model. We also present subquadratic algorithms for almost all other models.

The paper is organized as follows. In the next section we formally introduce the notation and the models that we study in the paper. Section 3 is devoted to the conditional center problem. We prove some nestedness results, showing that the solution to the conditional model when the new facility is restricted to be a point, is contained in an optimal solution to the problem of locating an extensive facility of positive length. We use this result to design efficient algorithms for the latter problem. In Section 4 we study the median model. We prove the above NP-hardness result and present the approximation algorithm. We also provide efficient algorithms for the path models.

2. The conditional subtree/path problem under the size constraint

Let $T = (V, E)$ be an undirected tree network with node set $V = \{v_1, \ldots, v_n\}$. 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 $p(v_j)$, the parent of v_j, be the node $v \in V$, closest to v_j, $v \neq v_j$ on $P[v_1, v_j]$, the path connecting v_j to v_1. v_j is a child of $p(v_j)$. We let e_j be the edge connecting v_j with its parent $p(v_j)$. Hence, the edge set is $E = \{e_2, \ldots, e_n\}$. If v_i, v_k are the two nodes of e_j, we will also use the notation $e_j = (v_i, v_k)$. A node v_i is a descendant of v_j if v_j is on $P[v_i, v_1]$. V_j will denote the set of all descendants of v_j.

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. We view $A(T)$ as a connected and closed set which is the union of $n - 1$ intervals. Let $P[v_i, v_j]$ denote the unique simple path in $A(T)$ connecting v_i and v_j.

We refer to interior points on an edge by their Euclidean 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 belong to the same edge we will refer to $P[x, y]$ as a *subedge* or a *partial edge*. $A(T)$ is a metric space with respect to the above distance function.

The path $P[x, y]$ can also be viewed as a collection of edges and at most two subedges (partial edges). $P(x, y)$ will denote the open path obtained from $P[x, y]$ by deleting the points x, y, and $P(x, y]$ will denote the half open path obtained from $P[x, y]$ by deleting the point x. Also, for any subset $Y \subseteq A(T)$, and x in $A(T)$ we define $d(x, Y) = d(Y, x) = \inf\{d(x, y)\colon\ y \in Y\}$. (If Y is empty, we set $d(x, Y) = \infty$.) 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* if all its (relative) boundary points are nodes of T. We call a subtree *almost discrete* if at most one of its (relative) boundary points is not a node of T. If Y is a subtree we define the length or size of Y, $L(Y)$, to be the sum of the lengths of its complete and partial edges.

In our model the nodes of the tree are viewed as demand points (customers), and each node $v_i \in V$ is associated with a nonnegative weight w_i. The set of potential servers consists of subtrees. Specifically, let C be a collection of subtrees with the property that each point $x \in A(T)$ is at least in one element of C. For example, C can be the set of all subtrees (paths).

Let D be a collection of discrete subtrees with the property that each node $v_i \in V$ is at least in one element of D.

We assume that there is a subset of nodes S, where centers (servers) are already established. For example, S may represent the node set of an existing facility, which by itself can be a subtree or even a forest. In our models the goal is to minimize some specific monotone functions of the service distances of the customers to their respective nearest centers. We establish only one server, a subtree Y in C (D), with $L(Y) \leqslant L$. Hence, the service distance of node v_i is $\min[d(v_i, Y), d(v_i, S)]$. When the subtree is selected from C we refer to the model as the conditional continuous model, and if it is chosen from D, it is called the conditional discrete model. If S is empty we call the model *unconditional*. We focus in this paper on two specific objective functions; the two most common in location theory.

In the conditional (w-weighted) center problem the objective is to minimize

$$F_c(Y) = \max_{i=1,\ldots,n} w_i \min\bigl[d(v_i, Y), d(v_i, S)\bigr].$$

In the conditional (w-weighted) median problem the objective is to minimize

$$F_m(Y) = \sum_{i=1}^{n} w_i \min\bigl[d(v_i, Y), d(v_i, S)\bigr].$$

If $w_i = 1$ for each $v_i \in V$, the above models are called unweighted.

The main goal of this paper is to study conditional extensive location problems. For comparison purposes we summarize in Tables 1–4 the best known results for the unconditional and conditional cases, as well as our own results for these models, which are identified by bold letters. We note that all the algorithmic and complexity results in the paper refer to the cases where C and D, defined above, are either the collections of all relevant subtrees or the collections of all relevant paths, depending on the case.

Table 1
Unconditional subtree

		Discrete		Continuous	
		Complexity	Ref.	Complexity	Ref.
Median (nestedness property) [19,20,31]	Weighted	NP-hard	[11]	$O(n)$	[29]
	Unweighted	NP-hard	[11]	$O(n)$	[29]
Center (nestedness property) [19,20,31]	Weighted	$\boldsymbol{O(n \log n)}$		$O(n \log n)$	[31]
	Unweighted	$O(n)$	[27]	$O(n)$	[27]

Bold letters indicate new results in the paper.

Table 2
Conditional subtree

		Discrete	Continuous
Median	Weighted	NP-hard [11]*	**NP-hard**
	Unweighted	NP-hard [11]*	**NP-hard**
Center	Weighted	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
	Unweighted	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$

Bold letters indicate new results in the paper.
* It follows from the unconditional case.

Table 3
Unconditional path

			Discrete		Continuous	
			Complexity	Ref.	Complexity	Ref.
Median (no nestedness property) [19,20]	Unweighted	Length	$O(n \log n)$	[1]	$O(n \log n \alpha(n))$	[1]
		No length	$O(n)$	[22]	$O(n)$	[22]
	Weighted	Length	$O(n \log n)$	[1]	$O(n \log n \alpha(n))$	[1]
		No length	$O(n)$	[2]	$O(n)$	[2]
Center (nestedness property) [19,20]	Unweighted	Length	$O(n)$	[32,34]	$O(n)$	[32]
		No length	$O(n)$	[13]	$O(n)$	[13]
	Weighted	Length	$\boldsymbol{O(n \log n)}$		$\boldsymbol{O(n \log n)}$	
		No length	$\boldsymbol{O(n \log n)}$		$\boldsymbol{O(n \log n)}$	

Bold letters indicate new results in the paper.

Table 4
Conditional path

		Discrete	Continuous
Weighted median **(no nestedness property)**	Length	$\boldsymbol{O(n \log^2 n)}$	$\boldsymbol{O(n^2)}$
	No length	$\boldsymbol{O(n \log^2 n)}$	$\boldsymbol{O(n \log^2 n)}$
Weighted center **(nestedness property)**	Length	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
	No length	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$

Bold letters indicate new results in the paper.

3. The conditional center subtree/path problem

3.1. Unconditional center problems

3.1.1. Nestedness results for subtree and path center problems on trees

Let L be a nonnegative real number. The unconditional center problem on C (D) is to find a subtree Y in C (D), with $L(Y) \leqslant L$, minimizing the objective

$$\max_{i=1,\ldots,n} w_i d(v_i, Y).$$

We assume without loss of generality that at least two of the node weights are positive.

Lemma 3.1. *Let x^* be the unique solution to the continuous weighted* 1*-center problem on T. Then for each $L \geqslant 0$, $Y^*(L)$, an optimal solution to the unconditional center problem on C, contains x^*.*

Proof. The proof follows from the result in [31] for the more general centdian model. (The centdian objective function is a convex combination of the sum (median) and maximum (center) functions.) A simple direct proof for the center problem is as follows.

Let r_1 be the optimal solution value to the 1-center problem, i.e., the solution value for the case when $L = 0$. Then, there exist a pair of nodes, v_i, v_j, such that $w_i d(v_i, x^*) = w_j d(v_j, x^*) = r_1$, and x^* is on $P[v_i, v_j]$. Let $Y \in C$ satisfy $L(Y) \leqslant L$. Since Y is connected, it follows that if x^* is not in Y, then

$$\max\big[w_i d(v_i, Y), w_j d(v_j, Y)\big] > \max\big[w_i d\big(v_i, x^*\big), w_j d\big(v_j, x^*\big)\big] = r_1.$$

Hence Y is not an optimal solution. □

Lemma 3.2. *Let v_k be a solution to the discrete weighted* 1*-center problem on T. Then for each $L \geqslant 0$, there exists an optimal solution to the unconditional center problem on D which contains v_k.*

Proof. If x^*, defined in Lemma 3.1, is a node the result follows from Lemma 3.1. Otherwise, there exists a node v_t, such that x^* is in the interior of the edge (v_k, v_t). Let r'_1 (r_1) denote the optimal value for the discrete (continuous) weighted 1-center problem. We clearly have $r_1 < r'_1$. Note that if v_k is not a unique solution, then the only other solution is v_t.

Let $V_{k,t}$ ($V_{t,k}$) be the node set of the connected component containing v_k (v_t), obtained from T by removing the edge (v_k, v_t).

Let v_i satisfy $r'_1 = w_i d(v_k, v_i)$. Since $r_1 < r'_1$, it follows that $v_i \in V_{t,k}$. (Otherwise it would yield the contradiction $r'_1 = w_i d(v_i, v_k) < w_i d(v_i, x^*) \leqslant r_1$.)

Since v_k is optimal there must be a node $v_j \in V_{k,t}$ such that $w_j d(v_j, v_t) \geqslant r'_1$. (Otherwise, it would follow that for each node $v_s \in V_{k,t}$, $w_s d(v_s, v_t) < r'_1$, and for each node $v_s \in V_{t,k}$ with $w_s > 0$, $w_s d(v_s, v_t) < w_s d(v_s, v_k) \leqslant r'_1$; implying that v_t is a better solution than v_k.)

Consider now an optimal solution Y of length $L > 0$ in D. Without loss of generality suppose that its objective value is smaller than r'_1. This implies that Y cannot be contained

in $V_{k,t}$ since it would imply $w_i d(v_i, Y) \geqslant w_i d(v_i, v_k) = r'_1$. Also, Y cannot be contained in $V_{t,k}$, since it would imply $w_j d(v_j, Y) \geqslant w_j d(v_j, v_t) \geqslant r'_1$. Hence, since Y is a discrete subtree, it must contain the entire edge (v_k, v_t). □

3.1.2. Algorithms for unconditional center problems

From the above nestedness results we conclude that to solve unconditional, continuous/discrete, subtree/path, weighted center problems with a length constraint on trees, it is sufficient to solve the rooted versions, where the subtree/path must contain some distinguished point. For convenience, we suppose without loss of generality that this distinguished point is v_1, the root of T.

The main ingredient in algorithms solving center problems is a feasibility test. Given a real number r, determine whether there exists a subtree/path, rooted at v_1, of length not exceeding L, such that the weighted distance of each node from the subtree/path is at most r. We now show how to perform this test in linear time.

The feasibility test

For each $v_i \in V$ define $r_i = r/w_i$. If $d(v_i, v_1) > r_i$, define x_i to be the point on $P[v_i, v_1]$ satisfying $d(x_i, v_i) = r_i$. Otherwise, set $x_i = v_1$. Also, define y_i to be the closest node to x_i on $P[v_i, x_i]$. Let $X' = \{x_i\colon\ v_i \in V\}$ and $Y' = \{y_i\colon\ v_i \in V\}$. Define the subsets of least elements $X'' = \{x_i \in X'\colon\ \nexists\, x_j \neq x_i, x_i \in P[x_j, v_1]\}$, and $Y'' = \{y_i \in Y'\colon\ \nexists\, y_j \neq y_i, y_i \in P[y_j, v_1]\}$.

It is clear that the test is positive for the continuous subtree (path) problem if and only if the length of the subtree (path) induced by X'' and the root v_1 is at most L. Similarly, the test is positive for the discrete subtree (path) problem if and only if the length of the subtree (path) induced by Y'' and the root v_1 is at most L. (Note that in the case of a path, if $|X''| > 2$ ($|Y''| > 2$), the respective test is negative.) Therefore to obtain an $O(n)$ feasibility test it is sufficient to show how to generate X'' and Y'' in linear time.

We describe an $O(n)$ procedure for finding X''. Initially, X'' is empty.

Select an arbitrary node v_j of the rooted tree such that all its children are leaves. Let $S(v_j)$ be the set of children of v_j. If $d(v_i, v_j) \leqslant r_i$ for each child $v_i \in S(v_j)$, replace r_j by $\min[r_j, \min_{v_i \in S(v_j)}[r_i - d(v_i, v_j)]]$. Delete all the edges (v_i, v_j), $v_i \in S(v_j)$. Proceed inductively. Otherwise, let v_t, a child of v_j with $d(v_t, v_j) > r_t$. Augment the point x_t to X''. Let $T_1, \ldots, T_q$ be the collection of connected components obtained from T by removing all edges on $P[v_t, v_1]$. (Each component T_m is a subtree rooted at some node, say $v_{j(m)}$ on $P[v_t, v_1]$.) Recursively, find the respective subsets of least elements $X''_1, \ldots, X''_q$ of these components. Then

$$X'' = \{x_t\} \cup \left\{X''_1 - \{v_{j(1)}\}\right\} \cup \cdots \cup \left\{X''_q - \{v_{j(q)}\}\right\}.$$

It is clear that the total complexity is linear. The above procedure can easily be modified to construct Y''. (Replace x_t by y_t, and $X''_1, \ldots, X''_q$ by $Y''_1, \ldots, Y''_q$.)

We are now ready to present $O(n \log n)$ time algorithms for the different versions of the unconditional center models. The general approach is to identify a set containing the optimal solution value, and then use the feasibility test to search for the optimal value in that set.

1. For the continuous subtree model we already have an $O(n \log n)$ time algorithm in [31]. (It works for the more general centdian problem.)
2. A set containing the optimal value for the continuous path model: The optimal value is an element in the set $R = R_1 \cup R_2$ where

$$\begin{aligned} R_1 &= \{w_i d(v_i, v_j)\colon\ v_j \in P[v_i, v_1],\ i, j = 1, \ldots, n\} \\ &= \{w_i[d(v_i, v_1) - d(v_j, v_1)]\colon\ i, j = 1, \ldots, n\}, \end{aligned}$$

$$R_2 = \{[w_i w_j/(w_i + w_j)][d(v_i, v_1) + d(v_j, v_1) - L]\colon\ i, j = 1, \ldots, n\}.$$

   Indeed, if $r \notin R_1$ then there must exist a balance between two nodes v_i, v_j. Assume that the closest nodes to each side of the optimal path P ($L(P) = L$) are v_k and v_t and the distances $d(P, v_k) = y$, $d(P, v_t) = x$. Then, the following two linear equations must hold:

$$w_i(d(v_i, v_k) + y) = w_j(d(v_j, v_t) + x), \tag{1}$$

$$L + x + y = d(v_k, v_t). \tag{2}$$

   The admissible radii are of the form $r = w_i(d(v_i, v_k) + y)$. If we substitute (1) and (2) into this equation we get:

$$\begin{aligned} r &= w_i\left(d(v_i, v_k) + \frac{w_j(d(v_k, v_j) - L) - w_i d(v_i, v_k)}{w_i + w_j}\right) \\ &= \frac{w_i}{w_i + w_j}\big(d(v_i, v_k)(w_i + w_j) + w_j(d(v_k, v_j) - L) - w_i d(v_i, v_k)\big) \\ &= \frac{w_i w_j}{w_i + w_j}(d(v_i, v_j) - L). \end{aligned}$$

3. For the continuous path model we get an $O(n \log n)$ algorithm by searching over the set R above. The search over the set R_2 is described in [31]. The search over the set R_1 is even simpler, and it is described in [16].
4. For the discrete subtree and path models the optimal value is definitely an element in the set R_1, defined above. Therefore these models are also solvable in $O(n \log n)$ time.

3.2. Conditional center problems

3.2.1. Nestedness results for subtree and path center problems on trees

We will next prove nestedness results, similar to Lemmas 3.1 and 3.2, for the conditional models. Unlike the unconditional case, as illustrated by the next example, for the conditional model, the nestedness property holds for some solution to the problem with $L = 0$, but not necessarily for all such solutions.

Example 3.1. Consider 4 points (nodes) on the real line: $(v_1, v_2, v_3, v_4) = (0, 1, 3.1, 4.1)$. Suppose that $S = \{v_1, v_4\}$, and $w_i = 1$, for $i = 1, 2, 3, 4$. Any point on $P[v_1, v_4]$ is optimal for the case $L = 0$. However, no point in $P[v_1, v_2]$ or $P[v_3, v_4]$ will satisfy the nestedness property of Lemma 3.3 when $0.1 < L < 1$. The property is satisfied at $y^* = 2.05$.

Fig. 1. Illustration of Example 3.1. Big dots represent the conditional set S.

Lemma 3.3. *There exists a solution, $y^* \in A(T)$, to the conditional continuous weighted center problem on C with $L = 0$, such that for any $L \geqslant 0$, there is an optimal solution, $Y^*(L)$, to the conditional center problem on C with $L(Y^*(L)) \leqslant L$, and $y^* \in Y^*(L)$.*

Proof. For each node v_i let $r'_i = w_i d(v_i, S)$. Define $r' = \max_{i=1,\ldots,n} r'_i$. Suppose without loss of generality that $r' > 0$. Let $x^* \in A(T)$ be an optimal solution to the conditional continuous weighted center problem on C with $L = 0$, and let r^* be the optimal solution value.

Suppose first that $r^* = r'$. Consider the set V', consisting of all nodes in V such that $r'_i = r'$. We clearly have $|V'| \geqslant 2$. Then without loss of generality x^* is the optimal solution for the (unconditional) weighted 1-center problem for the nodes in V'. In particular, there is a pair of nodes v_i, v_j in V' such that x^* is on $P[v_i, v_j]$, and $w_i d(v_i, x^*) = w_j d(v_j, x^*) \geqslant r'$. From the argument used in the proof of Lemma 3.1, we conclude that the result holds for $y^* = x^*$.

Suppose now that $r^* < r'$. Without loss of generality assume that $r^* > 0$, otherwise, the result clearly holds. Define the following subsets of V.

$$V_- = \{v_i\colon\ w_i d(v_i, S) < r^*\},$$
$$V_= = \{v_i\colon\ w_i d(v_i, S) = r^*\},$$
$$V_+ = \{v_i\colon\ w_i d(v_i, S) > r^*\}.$$

From the fact that $r^* < r'$, it follows that V_+ is nonempty. Moreover, the nodes in V_+ are served by x^*. Let $r'' = \max_{v_i \in V_+} w_i d(v_i, x^*)$. If $r'' = r^*$, then $r'' > 0$. We can now assume without loss of generality that x^* is the solution to the weighted 1-center problem for the nodes in V_+. Again, from the argument used in the proof of Lemma 3.1, we conclude that the result holds for $y^* = x^*$. Hence, it is sufficient to consider the case where $0 \leqslant r'' < r^*$.

Define $r_i = r^*/w_i$, for $i = 1, \ldots, n$. Let

$$T'' = \{x \in A(T)\colon\ d(x, v_i) \leqslant r_i,\ \forall v_i \in V_+\}.$$

Note that since $r'' < r^*$, T'' is a (neighborhood) subtree with nonempty interior. Moreover, each boundary point of T'', which is not a leaf of $A(T)$ is at a distance of r_i from some node $v_i \in V_+$.

Assume without loss of generality that $V_= = \{v_1, \ldots, v_k\}$. For $t = 1, \ldots, k$, let

$$T_t = \{x \in A(T)\colon\ d(x, v_t) \leqslant r_t\}.$$

If the intersection of T_1 and T'' is empty, then there is a boundary point y^* of T'', and a node $v_i \in V_+$, such that y^* is on $P[v_1, v_i]$, and $w_i d(v_i, y^*) = r^* = w_1 \min[d(v_1, y^*), d(v_1, S)]$. Therefore, for any $L > 0$, there is an optimal solution to the conditional center problem on C, with value smaller than r^*, only if this solution contains interior points on the subpaths $P[v_1, y^*]$ and $P[v_i, y^*]$. Hence the boundary point

y^* satisfies the nestedness property in the lemma. The same argument applies when the intersection of T_1 and T'' is a single point (which must be a boundary point of T''). If the intersection has a nonempty interior, we augment v_1 to V_+, and update T'' respectively. We then proceed by considering the intersection of T_2 with T'', etc.

We claim that we must reach a step where the intersection of T_t with the (updated) neighborhood subtree T'' is either empty or a singleton. (From the above argument this would conclude the proof.) If this were not the case we would conclude that the intersection of all neighborhood subtrees $T_i = \{x \in A(T)\colon\ d(x, v_i) \leqslant r_i\}$, $v_i \in V_+ \cup V_=$, has a nonempty interior. In particular, there is a point $x \in A(T)$ such that $w_i d(v_i, x) < r^*$, for all $v_i \in V_+ \cup V_=$, contradicting the optimality of r^*. □

Remark 3.2.1. The above proof suggests an $O(n \log n)$ algorithm for finding y^*. The complexity of the algorithm is determined by the effort to intersect $O(n)$ neighborhoods of a tree. Each intersection can be performed in $O(\log n)$ time, by implementing the formula for intersection in [9]. The intersection of two neighborhoods is by itself a neighborhood subtree. Its radius can be obtained from the radii of the two given neighborhoods in constant time. Its center is on the path connecting the centers of the two neighborhoods. Therefore, it can be found in $O(\log n)$ time by using the data structure in [16].

Lemma 3.4. *There exists a solution, $y^* \in V$, to the conditional discrete weighted center problem on D with $L = 0$, such that for any $L \geqslant 0$, there is an optimal solution, $Y^*(L)$, to the conditional discrete center problem on D with $L(Y^*(L)) \leqslant L$, and $y^* \in Y^*(L)$.*

Proof. We modify the proof of Lemma 3.3 for the discrete case.

For each node v_i let $r'_i = w_i d(v_i, S)$. Define $r' = \max_{i=1,\ldots,n} r'_i$. Suppose without loss of generality that $r' > 0$. Let $x^* \in V$ be an optimal solution to the conditional discrete weighted center problem on D with $L = 0$, and let r^* be the optimal solution value.

Suppose first that $r^* = r'$. Consider the set V', consisting of all nodes in V such that $r'_i = r'$. We clearly have $|V'| \geqslant 2$. Then without loss of generality x^* is an optimal solution for the discrete weighted 1-center problem for the nodes in V'. From the argument used in the proof of Lemma 3.2, we conclude that there is a pair of nodes $v_i, v_j \in V'$, such that x^* is on $P[v_i, v_j]$, and $r' \leqslant w_i d(v_i, x^*)$, $w_i d(v_i, x^*) \geqslant w_j d(v_j, x^*)$. Moreover, if for some $L > 0$, the optimal solution value is smaller than $r' = r^*$, then every optimal solution must contain an edge on $P[v_i, v_j]$ which is incident to x^*. Thus, we conclude that the result holds for $y^* = x^*$.

Suppose now that $r^* < r'$. Without loss of generality assume that $r^* > 0$, otherwise, the result clearly holds. Define the following subsets of V.

$$V_- = \{v_i\colon\ w_i d(v_i, S) < r^*\},$$
$$V_= = \{v_i\colon\ w_i d(v_i, S) = r^*\},$$
$$V_+ = \{v_i\colon\ w_i d(v_i, S) > r^*\}.$$

From the fact that $r^* < r'$, it follows that V_+ is nonempty. Moreover, the nodes in V_+ are served by x^*. Let $r'' = \max_{v_i \in V_+} w_i d(v_i, x^*)$. If $r'' = r^*$, then $r'' > 0$. We can now assume without loss of generality that x^* is the solution to the discrete weighted 1-center

problem for the nodes in V_+. Again, from the argument used in the proof of Lemma 3.2, we conclude that the result holds for $y^* = x^*$. Hence, it is sufficient to consider the case where $0 \leqslant r'' < r^*$.

Define $r_i = r^*/w_i$, for $i = 1, \ldots, n$. Let

$$T'' = \{x \in A(T)\colon\ d(x, v_i) \leqslant r_i,\ \forall v_i \in V_+\}.$$

Note that since $r'' < r^*$, T'' is a (neighborhood) subtree with nonempty interior. Moreover, each boundary point of T'', which is not a leaf of $A(T)$ is at a distance of r_i from some node $v_i \in V_+$. (Note that the boundary points of T'' are not necessarily nodes. However, T'' contains the node x^* in its interior.)

Assume without loss of generality that $V_= = \{v_1, \ldots, v_k\}$. For $t = 1, \ldots, k$, let

$$T_t = \{x \in A(T)\colon\ d(x, v_t) \leqslant r_t\}.$$

If the intersection of T_1 and T'' is empty, then there is a boundary point z^* of T'', and a node $v_i \in V_+$, such that z^* is on $P[v_1, v_i]$, and $w_i d(v_i, z^*) = r^* = w_1 \min[d(v_1, z^*), d(v_1, S)]$. (Note that z^* is not necessarily in V.) Let y^* be the closest node to z^* on $P[v_i, z^*]$. It is now easy to check that for any $L > 0$, there is an optimal solution to the conditional center problem on C, with value smaller than r^*, only if this solution contains the node y^*. The same argument applies when the intersection of T_1 and T'' is a single point (which must be some boundary point, z^* of T'').

If $T'' \cap T_1$ has a nonempty interior, we distinguish between two cases.

Case I. $T'' \cap T_1$ contains no node in its interior.

In this case v_1 is not in T''. Define z^* to be the closest (boundary) point to v_1 in T'', and proceed as above.

Case II. $T'' \cap T_1$ contains a node in its interior.

In this case $T'' \cap T_1$ contains a node, say v_j such that $d(v_i, v_j) < r_i$, for each $v_i \in V_+$, and also $d(v_1, v_j) < r_1$. We augment v_1 to V_+, and update T'' respectively. We then proceed by considering the intersection of T_2 with T'', etc.

We claim that we must reach a step where the intersection of T_t with the (updated) neighborhood subtree T'' contains no node in its interior. (From the above argument this would conclude the proof.) If this were not the case we would conclude that the intersection of all neighborhood subtrees $T_i = \{x \in A(T)\colon\ d(x, v_i) \leqslant r_i\}$, $v_i \in V_+ \cup V_=$, contains a node, say v_j, such that $w_i d(v_i, v_j) < r^*$, for all $v_i \in V_+ \cup V_=$, contradicting the optimality of r^*. □

Remark 3.2.2. Note that Remark 3.2.1 is also applicable here, so that the point conditional discrete weighted center problem with a nestedness property can be found in $O(n \log n)$ time.

3.2.2. Algorithms for conditional center problems

Based on the above nestedness results for the conditional center problems, we conclude that to solve conditional, continuous/discrete, subtree/path, weighted center problems with

a length constraint on trees, it is sufficient to consider the rooted versions, where the subtree/path must contain some distinguished point. For convenience, suppose without loss of generality that this point is v_1, the root of T.

It is easy to modify the feasibility test so that it will correctly resolve the test for the conditional models. Also, note that sets containing the optimal solution values for the conditional models are obtained by augmenting the set $R' = \{w_i d(v_i, S)\colon\ v_i \in V\}$ to the sets of the respective unconditional models. For example, a set containing the optimal solution value for the conditional continuous path center problem is $R_1 \cup R_2 \cup R'$.

In the preprocessing phase we compute $R' = \{w_i d(v_i, S)\colon\ v_i \in V\}$ in $O(n)$ time as follows. Starting from the leaves of the rooted tree, and proceeding recursively to the root, in linear time we can find, for each node v_j, its distance, say d_j, to the closest node of S in V_j. At the end of this phase we already have $d(v_1, S)$. In the second phase we start at the root and proceed recursively to the leaves, computing $d(v_j, S)$ from d_j and $d(p(v_j), S)$ in constant time. (Recall that $p(v_j)$ is the parent of v_j, and therefore $d(v_j, S) = \min[d_j, d(p(v_j), S) + d(v_j, p(v_j))]$.) Hence, in $O(n)$ time we compute $d(v_j, S)$ for all the nodes.

With the above information we can now mimic the solution approach for the unconditional models and solve the continuous and discrete conditional models in $O(n \log n)$ time.

An alternative solution strategy for a conditional model is to reduce it to a respective unconditional problem in $O(n \log n)$ time.

Using the above notation let r^* denote the optimal value of the conditional model. Let $I = (r''_1, r''_2, \ldots, r''_n)$ be the sorted list of the elements in R'. Our first task is to identify a pair of consecutive elements in I, say r''_t and r''_{t+1} such that $r''_t < r^* \leqslant r''_{t+1}$. We perform a binary search on R'. Select in linear time r''_s, a median element of R'. Let $V' = \{v_i \in V\colon\ w_i d(v_i, S) \leqslant r''_s\}$. Then $r^* \leqslant r''_s$, if and only if there is a subtree (path), whose length is at most L, such that the weighted distance of each node in $V - V'$ from the subtree (path) is at most r''_s. We can use the above feasibility test (for the unconditional model) on the nodes in $V - V'$ to resolve this query. We then continue the binary search, on R', and after $O(\log n)$ steps we identify (in $O(n \log n)$ time), the pair r''_t, r''_{t+1}, for the conditional continuous (discrete) center problem. To solve the conditional model it is now sufficient to solve the respective unconditional model only for customers in $\{v_i \in V\colon\ w_i d(v_i, S) \geqslant r''_{t+1}\}$. The total effort is clearly $O(n \log n)$.

4. The conditional median subtree/path problem

As we see in Table 1, unconditional median subtree problems (with a length constraint) are NP-hard for the discrete model and linearly solvable in the continuous case. Moreover, fully polynomial time approximation schemes (*FPTAS*) for the discrete case are given in [29]. In this section we show that the conditional subtree median problem is NP-hard even for the continuous case. We then demonstrate that both the discrete and the continuous conditional median subtree problems have FPTAS. These algorithms are simple modifications of the scheme given in [29] for the unconditional discrete case.

The unconditional and the conditional discrete median path problems are clearly polynomially solvable, since there is only a quadratic number of paths to be considered. Hence, $O(n^3)$ time algorithms are trivially available for all discrete problems. For the unconditional case (see Table 3), linear time algorithms are known when the size of the path is unrestricted. For the case when the length is bounded, a subquadratic algorithm is known only for the case where all edges have unit length and the node weights are identical [23]. We present subquadratic algorithms for all but one of the median path models.

Before we start the detailed discussion we observe that the continuous conditional median subtree and path problems are actually "almost" discrete. (See Section 2 for a formal definition.) Consider an edge (v_i, v_j), and let x be a point on the edge. Then for each node v_k the function $\min[d(v_k, x), d(v_k, S)]$ is clearly a concave function on the edge (v_i, v_j). Let $P[x, y]$ be a path of length L connecting a point x on (v_i, v_j) with a point y on (v_s, v_t). Suppose that v_j and v_s are on $P[x, y]$. (To simplify the notation suppose without loss of generality that $x = d(x, v_j)$ and $y = d(y, v_s)$.) Next consider the problem of finding the continuous path P, of length L, which has one endpoint in (v_i, v_j), the other in (v_s, v_t), and it minimizes $\sum_{k=1}^{n} w_k \min[d(v_k, P), d(v_k, S)]$. From the above this problem reduces to a minimization of a concave function of the two variables x and y, subject to the constraint that each one of them is restricted to an interval and their sum (total length) is constant. Therefore, we conclude that there is an optimal continuous path where one of its endpoints is a node. A similar observation clearly holds for an optimal continuous median subtree, since we can apply the above to any maximal subpath of a given subtree.

The next lemma summarizes the above.

Lemma 4.1. *For each $L \geqslant 0$ there is an optimal continuous conditional median subtree* (*path*) *of length L which is almost discrete.*

4.1. The conditional median subtree problem

As defined in the introduction above, the conditional (w-weighted) median subtree problem under the size constraint consists of locating a subtree, $Y \subseteq A(T)$ such that $L(Y) \leqslant L$ where there exist already servers at the subset of nodes S. The demand points (nodes) are allocated to the closest server, either Y or S minimizing the weighted sum of the distances. The problem can be formulated as follows:

$$\min_{Y \subseteq A(T)} F_m(Y) := \sum_{i=1}^{n} w_i \min\big[d(v_i, Y), d(v_i, S)\big],$$
$$\text{s.t. } L(Y) \leqslant L. \tag{CMS}$$

Imposing that a given node, say v_1 must belong to Y, this problem admits the following reformulation

$$\min_{Y \subseteq A(T)} \sum_{i=1}^{n} w_i \min\big[z_i, d(v_i, S)\big],$$
$$\text{s.t. } \sum_{j=2}^{n} l_j x_j \leqslant L,$$

Fig. 2. Illustration of Example 4.1. Big dots are the nodes in the conditional set S.

$$x_j(1-x_i)=0 \quad \text{if } v_i=p(v_j),\ j=2,\dots,n, \tag{3}$$

$$\sum_{v_k\in P[v_1,v_i]} l_k(1-x_k)=z_i, \quad i=2,\dots,n, \tag{4}$$

$$0\leqslant x_j\leqslant 1, \quad j=2,\dots,n.$$

(Notice that since S is known then the terms $\{d(v_i,S)\}$ can be obtained in a preprocessing phase and hence can be considered as data. We have already shown in the previous sections that the effort needed to compute these terms for all nodes is $O(n)$.) In the unconditional model S is empty, and the objective is replaced by $\sum_{i=1}^{n} w_i z_i$. It is shown in [29] that for the unconditional case constraint (3) can be removed from the formulation. Moreover, this leads to a linear time algorithm. As illustrated by Example 4.1, this constraint cannot be removed in the conditional case. (The solution to the relaxed problem does not induce a connected set of $A(T)$.)

Another desirable property that holds for the unconditional median subtree problem is nestedness [18,19]: There exists an optimum subtree of any positive length in the unconditional version of the problem which contains an optimal point solution. Example 4.2 illustrates that this may not hold for the conditional subtree/path model.

Example 4.1. Let $T=(V,E)$ be a tree, where the set of nodes and edges are given by $V=\{v_1,v_2,v_3,v_4\}$ and $E=\{(v_1,v_2),(v_2,v_3),(v_3,v_4)\}$. The nodes are points on the real line with $(v_1,v_2,v_3,v_4)=(0,5,10,15)$ and $w_i=1$ for $i=1,2,3,4$. We assume that servers are already located at $S=\{v_2,v_3\}$ (see Fig. 2). Let $L=10$ be the upper bound of the length of Y. For any tree of length 10, the minimum objective value that we can obtain is 5. However, we see that if we select the (disconnected) set $Y=\{(v_1,v_2),(v_3,v_4)\}$, then the value of the objective function is 0. Thus, connectivity (constraint (3)), must be imposed in the formulation.

Example 4.2. Consider a tree $T=(V,E)$ where $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7\}$ and $E=\{(v_1,v_2),(v_2,v_3),(v_2,v_4),(v_2,v_5),(v_5,v_6),(v_6,v_7)\}$. The embedding of the nodes in the plane is given by $v_1=(0,0)$, $v_2=(5,0)$, $v_3=(5,5)$, $v_4=(5,-6)$, $v_5=(25,0)$, $v_6=(30,0)$, $v_7=(35,0)$. Let $w_i=1$ for $i=1,\dots,7$. We assume that servers are already located at $S=\{v_2,v_5\}$ (see Fig. 3). Let $L=16$ be the upper bound of the length of Y. The optimal (conditional on S) median point solution consists of locating the facility at any point of the edge (v_6,v_7). However, the optimum subtree Y verifying that $L(Y)\leqslant 16$ is the tree spanned by the set of nodes to $V_Y=\{v_1,v_2,v_3,v_4\}$. The example can be easily modified for the case of path facility location. For that case just consider the set $S=\{v_1,v_2,v_5\}$.

In general, problem (CMS) is NP-hard.

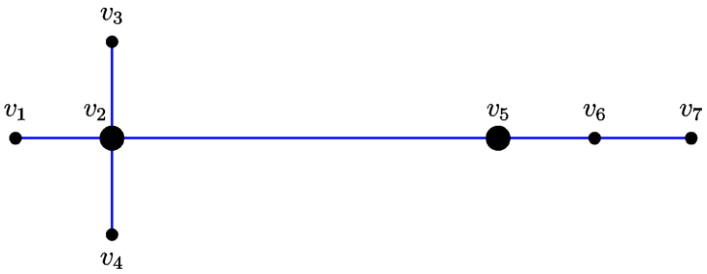

Fig. 3. Illustration of Example 4.2. Big dots are the nodes in the conditional set S.

Theorem 4.1. *The conditional continuous subtree median problem is NP-hard even for star graphs with all node weights being equal to* 1.

Proof. The following partition problem can be reduced to the conditional continuous subtree model: Given integers $a_1, \ldots, a_n$, is there a subset of them with sum equal to $A/2$, where A is the sum of all elements. Consider a star tree $T = (V, E)$ with $V = \{v_0, v_1, \ldots, v_n\}$ and $E = \{(v_0, v_1), \ldots, (v_0, v_n)\}$. The length of (v_0, v_i) is $2a_i$. Insert an extra node, u_i, at the middle of each edge (v_0, v_i). The modified star will have $2n + 1$ nodes, with v_0 as the center of the star. The existing facility (subtree) will consist of the edges (v_0, u_i), i.e., $S = \{v_0, u_1, \ldots, u_n\}$. Now let the length of the new tree be A.

It is now clear that there is an optimal solution to the location problem with value $A/2$ if and only if there is a solution to the above NP-hard partition problem. □

4.2. The $(1 + \varepsilon)$-approximation algorithm

From Table 1 we see that the unconditional median subtree problem is NP-hard for the discrete model and linearly solvable in the continuous case. Moreover, for the discrete case Tamir [29] presents an $(1 + \varepsilon)$-approximation algorithm. From the above results we know that in the conditional case even the continuous median subtree problem is NP-hard. Nevertheless, the fully polynomial approximation scheme in [29] can easily be modified for both, the discrete and continuous conditional median subtree model versions of problem (CMS). For the sake of brevity we give only a short description of the modification needed to obtain such an algorithm for the conditional discrete median subtree problem.

Given an instance of the problem and a positive ε, the algorithm generates in $O(n^3/\varepsilon)$ time, a subtree Y such that $L(Y) \leqslant L$ and $F(Y) \leqslant (1 + \varepsilon)F^{\text{opt}}$ where F^{opt} is the optimal solution of problem (CMS). The approach uses the interval partitioning method of [25]. First of all, we describe a pseudopolynomial time algorithm to solve the original problem. This is done using an adaptation of the Left–Right Dynamic Programming algorithm (L-R algorithm) described in [29] for solving the unconditional discrete problem. To apply this algorithm to our problem we must replace the distances $d(v_i, Y)$ by $\min[d(v_i, Y), d(v_i, S)]$.

To implement this efficiently, we start with the preprocessing, where for each node v_i we compute

$$A_i = \sum_{j=1}^{n} w_j \min\bigl[d(v_i, v_j), d(v_j, S)\bigr].$$

We define the following terms. If (v_i, v_j) is an edge, let $V_{j,i}$ be the set of all nodes v_k such that v_j is on the path connecting v_k to v_i. Define

$$a_{i,j} = \sum_{v_k \in V_{j,i}} w_k \min\bigl[d(v_i, v_k), d(v_k, S)\bigr].$$

It is then clear that

$$A_i = \sum_{v_j:\ (v_i,v_j)\in E} a_{i,j}.$$

We next show how to compute all these terms in $O(n \log^2 n)$ time. (We note that in the unconditional case all these terms are computable in $O(n)$ time [15]).

4.2.1. An $O(n \log^2 n)$ algorithm to compute $\{a_{i,j}\colon (v_i, v_j) \in E\}$

The algorithm has two phases. In the first phase we compute, in $O(n \log n)$ total time, all the terms $a_{i,j}$, where v_j is a child of v_i.

In the second phase we compute, in $O(n \log^2 n)$ time, all the terms A_i, $v_i \in V$. With the information from the first phase we can then also derive all the terms $a_{i,j}$, where v_j is the parent of v_i, in additional linear effort.

Phase I. In the previous section we showed how to compute in linear time $\{d(v_j, S)\colon v_j \in V\}$. For each node v_j, if $d(v_j, S) \geqslant d(v_j, v_1)$, define $x_j = v_1$, otherwise, define x_j to be the unique point on $P[v_j, v_1]$ such that $d(v_j, x_j) = d(v_j, S)$. It is shown in [15], that these points can be located in $O(n \log n)$ total time. We now use a bottom-up algorithm, starting at the leaves, to compute $a_{i,j}$ when v_j is a child of v_i.

Suppose that v_i is the parent of v_j. Define $U_j = \{v_k \in V_j\colon x_k \in P[v_k, v_j]\}$, and $U_{i,j} = \{v_k \in V_j\colon x_k \in P(v_j, v_i]\}$. Let

$$W_j = \sum_{v_k \in V_j} w_k \quad \text{and} \quad W'_j = \sum_{v_k \in U_j} w_k.$$

It is clear that the total effort to compute the terms $\{W_j\}$ and $\{W'_j\}$ is linear. Then

$$a_{i,j} = \sum_{v_t \in S(v_j)} a_{j,t} + \sum_{v_k \in U_{i,j}} w_k \bigl(d(v_k, S) - d(v_k, v_j)\bigr) + d(v_j, v_i)\left(W_j - W'_j - \sum_{v_k \in U_{i,j}} w_k\right).$$

From the above equation we conclude that the effort to calculate $a_{i,j}$ is proportional to $(|S(v_j)| + |U_{i,j}|)$. Therefore, in addition to the effort to locate the points $\{x_j\}$ we need $O(n)$ time to complete Phase I.

Phase II. In this phase we compute all the terms $\{A_i\}$ defined above in $O(n\log^2 n)$ time. (Note that when A_i is known we can compute $a_{i,j}$, for the case when v_i is a child of v_j, in $O(|S(v_i)|)$ time by the expression $a_{i,j} = A_i - \sum_{v_t\in S(v_i)} a_{i,t}$.)

The approach is very similar to the one described in Section 2.3.1 in [15]. It is based on divide and conquer. First we find in linear time a centroid of the tree, say v_j. From [16], we know that $T = (V, E)$ can be decomposed into two subtrees $T^1 = (V^1, E^1)$ and $T^2 = (V^2, E^2)$, such that $V^1 \cup V^2 = V$, $V^1 \cap V^2 = \{v_j\}$, $E^1 \cup E^2 = E$, and $n_p = |V^p| \leqslant 2(n+1)/3$, $p = 1, 2$.

For each node v_k define

$$B_k^1 = \sum_{v_t\in V^1} w_t \min\bigl[d(v_t, v_k), d(v_t, S)\bigr],$$

$$B_k^2 = \sum_{v_t\in V^2} w_t \min\bigl[d(v_t, v_k), d(v_t, S)\bigr],$$

$$C_k^1 = \sum_{v_t\in V^1-\{v_j\}} w_t \min\bigl[d(v_t, v_k), d(v_t, S)\bigr],$$

$$C_k^2 = \sum_{v_t\in V^2-\{v_j\}} w_t \min\bigl[d(v_t, v_k), d(v_t, S)\bigr].$$

Then, for each $v_k \in V^1$ ($v_k \in V^2$) we have $A_k = B_k^1 + C_k^2$ ($A_k = B_k^2 + C_k^1$). Due to the symmetry between V^1 and V^2, we show only how to compute the terms $\{A_k\}$ for all nodes $v_k \in V^1$.

We start by computing C_k^2 for all $v_k \in V^1$. Let $U^1 = (v_{i(1)}, \ldots, v_{i(n_1)})$ be the ordering of the nodes in V^1 by their distances from the centroid v_j. ($v_{i(1)} = v_j$.) Also define

$$V_*^2 = \bigl\{v_t \in V^2\colon\ d(v_t, v_j) > d(v_t, S)\bigr\} \quad \text{and} \quad C_*^2 = \sum_{v_t\in V_*^2} w_t d(v_t, S).$$

For each $v_t \in V^2 - V_*^2$ define $c_t = d(v_t, S) - d(v_t, v_j)$, and let

$$W^2 = \bigl(v_{q(1)}, \ldots, v_{q(n_2')}\bigr)$$

be the ordering of the nodes in $V^2 - V_*^2 - \{v_j\}$ by the keys $\{c_t\}$. ($n_2' = |V^2 - V_*^2 - \{v_j\}|$.) The total effort needed to generate U^1 and W^2 is clearly dominated by the sorting, and therefore it is $O(n\log n)$.

From the definition of U^1, we clearly have monotonicity, $C_{i(1)}^2 \leqslant \cdots \leqslant C_{i(n_1)}^2$. Moreover, to compute $C_{i(s)}^2$, for $s = 1, \ldots, n_1$, we only need to find the largest index $m = m(s)$, such that $d(v_{q(m)}, v_{i(s)}) \geqslant d(v_{q(m)}, S)$. (Note that the latter inequality is equivalent to $d(v_j, v_{i(s)}) \geqslant c_{q(m)}$. Therefore, $m(s)$ is monotone in s.) We have

$$C_{i(s)}^2 = C_*^2 + \sum_{r\leqslant m(s)} w_{q(r)} d(v_{q(r)}, S) + \sum_{r>m(s)} w_{q(r)} d(v_{q(r)}, v_{i(s)}).$$

From the monotonicity of $m(s)$ it follows that the additional time needed to compute $C_{i(s)}^2$ for all $s = 1, \ldots, n_1$, is $O(n)$. We conclude that the total time to compute C_k^2 for all nodes $v_k \in V^1$ is $O(n\log n)$. To compute $A_k = B_k^1 + C_k^2$ for all nodes $v_k \in V^1$ it is

now sufficient to compute B_k^1 for $v_k \in V^1$. The latter step is done recursively on the tree $T^1 = (V^1, E^1)$. (A symmetric procedure is applied to compute A_k for all $v_k \in V^2$.)

To evaluate the total effort needed let $C(n)$ denote the effort to compute the terms $\{A_k\}$ in a tree with n nodes. From the above discussion we obtain

$$C(n) \leqslant cn \log n + C(n_1) + C(n_2),$$

where $n_1 + n_2 = n + 1$, $n_1 \leqslant 2(n+1)/3$ and $n_2 \leqslant 2(n+1)/3$. We conclude that the total complexity is $C(n) = O(n \log^2 n)$.

4.2.2. The approximation algorithm

We now briefly describe the pseudopolynomial time algorithm.

Since v_1 is the root of $T = (V, E)$, we let $v_1, v_2, \ldots, v_n$ be a depth-first ordering of the nodes in V. Let D' be some known precomputed upper bound for the objective value of the problem (CMS). For each pair $[j, t]$, let $T'[j, t]$ be the subtree of T induced by all the nodes with indices lower than v_j plus the node v_j, the first t children of v_j (in order of index) and all the descendants of these t children.

We consider first the rooted version of problem (CMS), where the selected (discrete) subtree must contain the root v_1. The L-R algorithm maintains a sorted list $G[j, t]$ of pairs $(g[j, t, l], l)$ where $g[j, t, l]$ is the optimal solution of the problem in $T'[j, t]$ with length $l \leqslant L$ and $g[j, t, l] \leqslant D'$. (Since g is a nonincreasing function of l the ordering is well defined). The list only contains nondominated pairs, thus its order is $O(\min[L, D'])$. The optimal value of problem (CMS) is given by the smallest first (g) component of a pair in the list $G[n, 0]$.

In [29], it is proved that the time to compute a list $G[j, t]$ from a list $G[j, t-1]$ is $O(\min[L, D'])$. (For this update step we need to add the term $a_{j,j(t)}$, computed in the preprocessing, to the first coordinate of each pair in the list $G[j, t-1]$. $v_{j(t)}$ is the tth child of v_j.) Therefore, the total time to solve the problem by this algorithm is $O(n \min[L, D'])$. From the results in Section 3.2.2 we can compute, in $O(n \log n)$ time, a value of D' that is at most n times the optimal value of problem (CMS). Indeed, let Y_c^* be the optimal solution to the conditional center subtree problem, i.e., the solution to the minimax problem. Therefore, $F_M(Y_c^*)$ is clearly an n-approximation for the respective median model, which is problem (CMS). We have proved above that this n-approximation solution can be found in $O(n \log n)$ time. Thus, problem (CMS) can be solved in $O(n \min[L, nF^{\text{opt}}])$ time, where F^{opt} is the optimal value of problem (CMS).

We now sketch the fully polynomial time algorithm. Let $F^0 := F_M(Y_c^*)$ be the n-approximation given by the minimax solution, i.e., $F^0 \leqslant nF^{\text{opt}}$.

Given a positive ε, we partition the interval $[0, F^0]$ into $\lceil n^2/\varepsilon \rceil$ consecutive intervals, each but possibly the last of length $\lceil \varepsilon F^0/n^2 \rceil$.

The approximation algorithm follows the steps of the L-R algorithm. For each pair $[j, t]$ it produces a list $H[j, t]$ of at most $\lceil n^2/\varepsilon \rceil$ subtrees. Each subtree Y will be recorded by the pair $(F(Y), L(Y))$. The algorithm terminates with the final list $H[n, 0]$ that corresponds to the leaf node v_n.

The claim is that if (F^*, L^*), associated with the subtree Y^*, belongs to the list $H[n, 0]$ and is such that F^* is the smallest F coordinate (the first coordinate of the pairs (F, l)), then Y^* is a $(1+\varepsilon)$-approximation solution, i.e., $F^* \leqslant (1+\varepsilon)F^{\text{opt}}$.

Considering the (exact) pseudopolynomial algorithm, we note that there are only $O(n)$ subproblems $[j,t]$ and lists $G[j,t]$. Tamir [29] proved that if a list $G[j,t]$ is processed in the kth step of the algorithm and $(F(Y), L(Y))$ is one of the entrees of the list which is relevant for the optimal solution, then it is represented by some pair (F,l) in the list $H[j,t]$ where $|F(Y) - F| \leqslant k\varepsilon F^0/n^2$ and $L(Y) \leqslant l$. (At each step of the algorithm an additive error term $\varepsilon F^0/n^2$ is introduced.)

Therefore, for any pair (F,l) in the list $H[n,0]$ we have $|F - F^{\text{opt}}| \leqslant n\varepsilon F^0/n^2$. Since $F^0 \leqslant nF^{\text{opt}}$ we conclude that $F \leqslant F^{\text{opt}}(1+\varepsilon)$.

The above approach gives an $O(n^3/\varepsilon)$ time algorithm to obtain a $(1+\varepsilon)$-approximation solution to problem (CMS), where the selected subtree is rooted at v_1. This implies that the naive implementation to solve the unrooted version should solve n rooted subproblems. The total complexity would be $O(n^4/\varepsilon)$.

Again, following [29] there is a better implementation following a divide and conquer approach. Suppose without loss of generality that v_1 is a centroid of T. If v_1 is not included in the optimal subtree, it is included in a component having at most $n/2$ nodes. Hence, it is sufficient to approximate the problem where the optimal subtree must include v_1, and then make recursive calls to problems of size at most $n/2+1$. This analysis implies that the overall effort of obtaining a $(1+\varepsilon)$-approximation for the unrooted version of problem (CMS) is again $O(n^3/\varepsilon)$.

4.3. The conditional median path problem

Table 3 summarizes the best known results for unconditional median path problems. For the problems with a constraint on the length of the path, Alstrup et al. [1] give subquadratic algorithms. We will next give alternative short descriptions of slightly inferior algorithms for the sake of completeness. These algorithms use some preprocessing presenting in earlier sections. Then, we extend the algorithms to the conditional cases.

4.3.1. An $O(n\log n)$ algorithm for the discrete unconditional median path with a length constraint

We use a divide and conquer approach. First we find, in linear time, a centroid of the tree, say v_j. From [16], we know that $T=(V,E)$ can be decomposed into two subtrees $T^1=(V^1,E^1)$ and $T^2=(V^2,E^2)$, such that $V^1\cup V^2=V$, $V^1\cap V^2=\{v_j\}$, $E^1\cup E^2=E$, and $n_p=|V^p|\leqslant 2(n+1)/3$, $p=1,2$.

If an optimal path does not contain the centroid v_j, then it must be included either in T^1 or in T^2. Therefore, we can use a divide and conquer scheme. Find the best path containing nodes in both $V^1-\{v_j\}$ and $V^2-\{v_j\}$. Then, recursively find the best path contained in T^1 and the best path contained in T^2.

We start with a preprocessing phase described in [16], and modified by Frederickson and Johnson [10]. In this phase we find a centroid decomposition of the tree into a nested sequence of subtrees. For each subtree T' in this decomposition we compute recursively and sort the distances from its centroid, say v' to all other nodes of T'. The total time needed for this phase is $O(n\log n)$.

Suppose now that v_j is a centroid of the original tree. Our task is to find the best path containing nodes in both $V^1 - \{v_j\}$ and $V^2 - \{v_j\}$. For each node $v_k \in V^1$ ($v_k \in V^2$) let

$$A_k^1 = \sum_{v_t \in V^1} w_t d\big(v_t, P[v_j, v_k]\big) \quad \left(A_k^2 = \sum_{v_t \in V^2} w_t d\big(v_t, P[v_j, v_k]\big)\right).$$

Note, that A_k^1 (A_k^2) is the sum of the weighted distances of all the nodes in V^1 (V^2) from the path $P[v_j, v_k]$, connecting the centroid v_j with v_k. In linear time we compute $\{A_k^1\}$ and $\{A_k^2\}$, using the following equations.

If v_k is a node adjacent to v_t on the path connecting the centroid v_j to v_t in V^1 then,

$$A_t^1 = A_k^1 - a_{k,t} + A_t - a_{t,k}.$$

Similarly, if v_k is a node adjacent to v_t on the path connecting the centroid v_j to v_t in V^2 then,

$$A_t^2 = A_k^2 - a_{k,t} + A_t - a_{t,k}.$$

Let $U^1 = (v_{i(1)}, \ldots, v_{i(n_1)})$ be the ordering of the nodes in V^1 by their distances from the centroid v_j. Let $U^2 = (v_{q(1)}, \ldots, v_{q(n_2)})$ be the ordering of the nodes in V^2 by their distances from the centroid v_j. In particular $v_{i(1)} = v_{q(1)} = v_j$. We are now ready to compute the best discrete path of length not exceeding L which contains nodes in both $V^1 - \{v_j\}$ and $V^2 - \{v_j\}$.

It is sufficient to find, for each node v in U^1, the best path, whose length is at most L which has v as one of its endpoints. We start with $v_{i(n_1)}$. If $d(v_j, v_{i(n_1)}) > L$, there is no such path. Otherwise, find the largest index $t = t(n_1)$, such that $d(v_{q(t)}, v_{i(n_1)}) \leqslant L$, and $d(v_{q(t+1)}, v_{i(n_1)}) > L$. Set

$$\alpha_{i(n_1)} = A_{i(n_1)}^1 + \min_{s=1,\ldots,t(n_1)} A_{q(s)}^2,$$

where $\alpha_{i(n_1)}$ is the value of the best path which has $v_{i(n_1)}$ as one of its endpoints. Next we proceed with $v_{i(n_1-1)}$, and find the largest index $t = t(n_1 - 1)$ such that $d(v_{q(t)}, v_{i(n_1-1)}) \leqslant L$, and $d(v_{q(t+1)}, v_{i(n_1-1)}) > L$. It is clear that $t(n_1) \leqslant t(n_1 - 1)$. We then set

$$\alpha_{i(n_1-1)} = A_{i(n_1-1)}^1 + \min_{s=1,\ldots,t(n_1-1)} A_{q(s)}^2.$$

Continuing with $v_{i(n_1-2)}$ etc., in linear time we compute all the terms $\alpha_{i(r)}$, for $r = 1, \ldots, n_1$. We conclude that the objective value of the best discrete path of length not exceeding L which contains nodes in both $V^1 - \{v_j\}$ and $V^2 - \{v_j\}$ is given by $\min_{r=1,\ldots,n_1} \alpha_{i(r)}$.

In the recursive step we now have to find the best path contained in T^1 (T^2). (Due to symmetry we show only how to compute the path contained in T^1.) Consider a node $v_t \in V^2$. If P is some path contained in T^1, then $w_t d(v_t, P) = w_t d(v_t, v_j) + w_t d(v_j, P)$. Therefore, in order to solve the problem where the path is restricted to T^1, it is sufficient to replace the weight of v_j by $\sum_{v_t \in V^2} w_t$, remove all the nodes in $V^2 - \{v_j\}$ from T, and solve the problem on the remaining subtree, i.e., T^1. (Of course, we need to add the constant $\sum_{v_t \in V^2} w_t d(v_t, v_j)$ to the objective value of the restricted problem, to get the best objective value amongst all discrete paths contained in T^1.)

To evaluate the total effort needed to solve the problem recursively, let $C(n)$ denote the effort to compute the unconditional discrete median path of length not exceeding L in a tree with n nodes. We obtain

$$C(n) \leqslant cn + C(n_1) + C(n_2),$$

where $n_1 + n_2 = n + 1$, $n_1 \leqslant 2(n+1)/3$ and $n_2 \leqslant 2(n+1)/3$. We conclude that the total complexity is $C(n) = O(n \log n)$.

4.3.2. An $O(n \log^2 n)$ algorithm for the continuous unconditional median path with a length constraint

To solve the continuous median path problem we first recall that there is an optimal path such that one of its endpoints is a node. Using this property we can apply the same approach used for the discrete path. Given the notation of the previous section, we only need to show how to find the best path of length not exceeding L which contains a node in $V^1 - \{v_j\}$ ($V^2 - \{v_j\}$), and some point which is not v_j in T^2 (T^1). Due to symmetry we will consider only the paths which have a node in V^1 as one of their endpoints.

Consider the set $\{A^2_{q(s)}\}$, $s = 1, \ldots, n_2$. Let P be a path with an endpoint at some node $v_k \in V^1$. Its other endpoint is at a point $x_{i,m}$ on an edge (v_i, v_m) in T^2. (Suppose that v_i is on the path connecting v_m to the centroid v_j.) It is easy to see that the objective value of the path P is

$$A^1_k + A^2_i - \left(A^2_i - A^2_m\right)\left(d(v_i, x_{i,m}) \big/ d(v_i, v_m)\right).$$

Hence, the objective value of P varies linearly with the location of its endpoint $x_{i,m}$ on (v_i, v_m). Moreover, consider a path $P[v_j, v_t]$ connecting the centroid v_j with some leaf node v_t in V^2. If $y_{j,t}$ is a point on this path, and P is a path with $v_k \in V^1$ and $y_{j,t}$ as its two endpoints, the objective value of P is a monotone piecewise linear convex function of the location of $y_{j,t}$ on this path. The breakpoints of this function are the nodes of $P[v_j, v_t]$. Specifically, there is a piecewise linear function $f_{j,t}(y)$, of a real parameter y, $0 \leqslant y \leqslant d(v_j, v_t)$ such that for each node $v_k \in V^1$, and a point x on $P[v_j, v_t]$ satisfying $d(x, v_j) = y$, the objective value of the path $P[v_k, x]$ is $A^1_k + f_{j,t}(y)$. (Note that $f_{j,t}(0) = A^2_j$ and $f_{j,t}(d(v_j, v_t)) = A^2_t$.) For convenience, we extend the definition of $f_{j,t}(y)$ for all nonnegative values of y, by defining $f_{j,t}(y) = A^2_t$ for all $y \geqslant d(v_j, v_t)$. Let V^{2*} be the set of leaves of V^2. Define

$$F(y) = \min_{v_t \in V^{2*}} f_{j,t}(y).$$

Since the total number of breakpoints of all the functions $\{f_{j,t}\}$ is at most n_2, it is known that the total number of breakpoints of F is at most $O(n_2 \alpha(n_2))$, where $\alpha(n_2)$ is the inverse of the Ackermann function (see [26]). Moreover, the sequence of breakpoints of F can be generated in $O(n_2 \log n_2)$ time (see [14]).

We are now ready to compute, for each node $v_k \in V^1$, the best path whose length is at most L, which has one of its endpoints at v_k and the other at some point in T^2. By the above analysis the objective value of such a path is

$$A^1_k + F\left(L - d(v_k, v_j)\right).$$

$F(L - d(v_k, v_j))$ can be computed in $O(\log n_2)$ time by applying a binary search over the breakpoints of F. We now conclude that in $O(n \log n)$ time we can find the best path of length not exceeding L, which contains a node in $V^1 - \{v_j\}$ ($V^2 - \{v_j\}$) and some point, which is not v_j in T^2 (T^1).

As in the previous section we continue recursively with the subtrees T^1 and T^2. To evaluate the total effort needed to solve the problem recursively, let $C(n)$ denote the effort to compute the unconditional continuous median path of length not exceeding L in a tree with n nodes. We obtain

$$C(n) \leqslant cn \log n + C(n_1) + C(n_2),$$

where $n_1 + n_2 = n + 1$, $n_1 \leqslant 2(n+1)/3$ and $n_2 \leqslant 2(n+1)/3$. We conclude that the total complexity is $C(n) = O(n \log^2 n)$.

In the next subsections we describe efficient algorithms for the conditional models. We start with the $O(n \log^2 n)$ preprocessing phase, mentioned in the previous section, where we compute the terms $\{a_{i,j}\}$ for all edges (v_i, v_j).

4.3.3. An $O(n \log^2 n)$ algorithm for the conditional median path problem with no length constraint

When there is no length constraint, the conditional median path is a path connecting two leaves of the tree. Indeed, this problem can be solved in $O(n)$ time after all the terms $\{a_{i,j}\}$ have already been computed in $O(n \log^2 n)$ time. The approach is similar to that of [2]. Specifically, for each node v_i, we compute the optimal median path, which has v_i as one of its endpoints, and is contained in V_i. Let B_i denote the objective value of such an optimal path. Then recursively we have the following: If v_i is a leaf then $B_i = A_i$. Otherwise,

$$B_i = A_i + \min_{v_j \in S(v_i)} [B_j - a_{j,i} - a_{i,j}].$$

Finally to find the conditional (discrete) optimal median path with no length constraint, for each node v_i we compute the best path, which is contained in V_i, and contains v_i. Let C_i denote the objective value of such a path. Then we have the following: If v_i is a leaf then $C_i = B_i$. If v_i has only one child then again $C_i = B_i$. Suppose that $|S(v_i)| \geqslant 2$. Consider the set $\{B_j - a_{i,j} - a_{j,i}\colon\ v_j \in S(v_i)\}$. Let $j(1)$ and $j(2)$ be the indices corresponding to the two smallest entries in this set. Then it is easy to see that

$$C_i = A_i + B_{j(1)} - a_{j(1),i} - a_{i,j(1)} + B_{j(2)} - a_{j(2),i} - a_{i,j(2)}.$$

The objective value of the conditional optimal median path without length constraint is then $\min_{v_i \in V} C_i$. We therefore conclude that the conditional median path problem without length constraint can be solved in $O(n \log^2 n)$ time.

4.3.4. An $O(n \log^2 n)$ algorithm for the discrete conditional median path problem with a length constraint

We show how to adapt the $O(n \log n)$ algorithm from Section 4.3.1, which solves the respective unconditional model. We assume that all the $\{a_{i,j}\}$ coefficients have already been computed. We follow the notation in Section 4.3.1. Suppose now that v_j is a centroid

of the original tree. Our task is to find the best path containing nodes in both $V^1 - \{v_j\}$ and $V^2 - \{v_j\}$. For each node $v_k \in V^1$ ($v_k \in V^2$) let

$$A_k^1 = \sum_{v_t \in V^1} w_t \min\big[d\big(v_t, P[v_j, v_k]\big), d(v_t, S)\big]$$

$$\left(A_k^2 = \sum_{v_t \in V^2} w_t \min\big[d\big(v_t, P[v_j, v_k]\big), d(v_t, S)\big]\right).$$

Note, that A_k^1 (A_k^2) is the sum of the weighted distances of all the nodes in V^1 (V^2) from $P[v_j, v_k] \cup S$, where $P[v_j, v_k]$ is the path connecting the centroid v_j with v_k. In linear time we compute $\{A_k^1\}$ and $\{A_k^2\}$, using the following equations: Let V_j^1 (V_j^2) be the subset of V^1 (V^2) consisting of all the nodes adjacent to the centroid v_j in V^1 (V^2). Then,

$$A_j^1 = \sum_{v_t \in V_j^1} a_{j,t} \quad \text{and} \quad A_j^2 = \sum_{v_t \in V_j^2} a_{j,t}.$$

If v_k is a node adjacent to v_t on the path connecting the centroid v_j to v_t in V^1 then,

$$A_t^1 = A_k^1 - a_{k,t} + A_t - a_{t,k}.$$

Similarly, if v_k is a node adjacent to v_t on the path connecting the centroid v_j to v_t in V^2 then,

$$A_t^2 = A_k^2 - a_{k,t} + A_t - a_{t,k}.$$

Using the above expressions we proceed exactly as in Section 4.3.1, and find in linear time the objective value of the best discrete conditional median path of length not exceeding L, which contains nodes in both $V^1 - \{v_j\}$ and $V^2 - \{v_j\}$.

In the recursive step we now have to find the best path contained in T^1 (T^2). (Due to symmetry we show only how to compute the path in T^1.) We augment a node v_0 to V^1 and connect it with an edge to the centroid v_j. We define $a_{j,0} = A_j^2$, and solve the problem recursively on the augmented tree T^1. As in Section 4.3.1 the complexity of the recursive algorithm is $O(n \log n)$. However, in the conditional model the preprocessing phase of computing $\{a_{i,j}\}$ takes $O(n \log^2 n)$ time and determines the total complexity.

4.3.5. An $O(n^2)$ algorithm for the continuous conditional median path problem with a length constraint

At this stage we still do not know how to apply the above divide and conquer approach to the continuous conditional median path problem. Specifically, it is not clear to us how to aggregate the data from T^2 (T^1) into the centroid and decompose the problem into two "independent" subproblems on T^1 and T^2.

Instead, we use a direct approach to obtain an $O(n^2)$ algorithm. Since we know how to compute the best discrete path of length not exceeding L, in $O(n \log^2 n)$ time, we can assume, without loss of generality, that there exists an optimal almost discrete path P for the continuous problem, whose length is exactly L, and one of its endpoints, say x, is not a node. To obtain a quadratic time algorithm it is sufficient to restrict x to a given edge, and show how to get the best path with one endpoint on this edge in linear time.

First, we need some preprocessing. For each node v_k define

$$X_k = \{x \in A(T)\colon\ d(v_k, x) = d(v_k, S)\}.$$

Let $X = \bigcup_{k=1}^{n} X_k$. Note that $|X| = O(n^2)$. It is shown in [15] how to compute the points in X, and locate and sort them on the respective edges, in $O(n^2)$ time. We also compute and sort the distances from each node to all other nodes. This can also be performed in $O(n^2)$ time, as shown in [15].

We now consider an individual edge (v_i, v_j), and show how to find the best median path with an endpoint x, on this edge. (To simplify the notation we assume that $x = d(x, v_j)$.) Let $X_{i,j}$ denote the sorted list of all points of X on (v_i, v_j). We also augment the nodes v_i and v_j to this list. As above we let $V_{i,j}$ ($V_{j,i}$) be the set of nodes in the connected component containing v_i (v_j), obtained by removing the edge (v_i, v_j). For each x on (v_i, v_j) we define

$$g_{i,j}(x) = \sum_{v_t \in V_{i,j}} w_t \min\bigl[d(x, v_t), d(v_t, S)\bigr].$$

As noted, $g_{i,j}(x)$ is a monotone, piecewise linear and concave function with breakpoints at $X_{i,j}$. It is clear that in $O(n)$ total time we can compute $g_{i,j}(y)$, and $g_{j,i}(y)$, for all $y \in X_{i,j}$. Let x be a point on (v_i, v_j). It is sufficient to look only at paths of the type $P[v_k, x]$ of length L, where $v_k \in V_{j,i}$. For such a path the objective value is a piecewise linear concave function of x. Its breakpoints are in $X_{i,j}$. Similar to the notation in Section 4.3.4 we define $A_{k,j} = \sum_{v_t \in V_{j,i}} w_t \min[d(v_t, P[v_k, v_j]), d(v_t, S)]$. As explained there, in linear time we can compute these terms for all $v_k \in V_{j,i}$. The objective value of a path $P[v_k, x]$ is equal to $A_{k,j} + g_{i,j}(x)$, where $d(v_k, v_j) + x = L$.

Next we let $Z_{j,i} = (v_{k(1)}, \ldots, v_{k(n_{j,i})})$ be the ordering of the nodes in $V_{j,i}$ by their distances from v_j. ($v_{k(1)} = v_j$, and $n_{j,i} = |V_{j,i}|$.) Finally, by scanning $Z_{j,i}$ and the sequence of breakpoints $X_{i,j}$ we compute in $O(n)$ time the objective values of all paths of length L which have one endpoint in $V_{j,i}$ and the other end on (v_i, v_j).

We conclude that the total time to solve the continuous conditional median path problem with a length constraint is $O(n^2)$.

5. Final comments

We conjecture that the complexities of the algorithms for the conditional median path problems presented above can be further improved by using the data structures implemented by Alstrup et al. [1] to solve the unconditional versions of these models. This will be a subject of future research.

Acknowledgments

The research of the second and fourth authors was partially supported by Spanish MCyT grant numbers BFM2001-2378, HA2003-0121, BFM2004-0909. The third author

was supported by Spanish MCyT grant numbers BFM2000-1052-C02-01 and BFM2003-04062.

References

[1] S. Alstrup, P.W. Lauridsen, P. Sommerlund, M. Thorup, Finding cores of limited length, in: F. Dehne, A. Rau-Chaplin, J.-R. Sack, R. Tamassia (Eds.), Algorithms and Data Structures, in: Lecture Notes in Comput. Sci., vol. 1272, Springer, Berlin, 1997, pp. 45–54.
[2] I. Averbakh, O. Berman, Algorithms for path medi-centers of a tree, Comput. Oper. Res. 26 (1999) 1395–1409.
[3] R.I. Becker, Y. Perl, Finding the two-core of a tree, Discrete Appl. Math. 11 (1985) 103–113.
[4] O. Berman, D. Simchi-Levi, Conditional location problems on networks, Transport. Sci. 24 (1990) 77–78.
[5] R. Chen, Conditional minisum and minimax location–allocation problems in Euclidean space, Transport. Sci. 23 (1988) 157–160.
[6] R. Chen, G.Y. Handler, The conditional p-center problem in the plane, Naval Res. Logist. 40 (1993) 117–127.
[7] Z. Drezner, Conditional p-center problems, Transport. Sci. 23 (1989) 51–53.
[8] Z. Drezner, On the conditional p-median problem, Comput. Oper. Res. 22 (1995) 525–530.
[9] R.L. Francis, T.J. Lowe, H.D. Ratliff, Distance constraints for tree network multifacility location problems, Oper. Res. 26 (1978) 570–596.
[10] G.N. Frederickson, D.B. Johnson, Finding kth paths and p-centers by generating and searching good data structures, J. Algorithms 4 (1983) 61–80.
[11] S.L. Hakimi, E.F. Schmeichel, M. Labbé, On locating path—or tree shaped facilities on networks, Networks 23 (1993) 543–555.
[12] P. Hansen, M. Labbé, The continuous p-median of a network, Networks 19 (1989) 595–600.
[13] S.M. Hedetniemi, E.J. Cockaine, S.T. Hedetniemi, Linear algorithms for finding the Jordan center and path center of a tree, Transport. Sci. 15 (1981) 98–114.
[14] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, Inform. Process. Lett. 33 (1989) 169–174.
[15] T.U. Kim, T.J. Lowe, A. Tamir, J.E. Ward, On the location of a tree-shaped facility, Networks 28 (1996) 167–175.
[16] N. Megiddo, A. Tamir, E. Zemel, R. Chandrasekaran, An $O(n \log^2 n)$ algorithm for the kth longest path in a tree with applications to location problems, SIAM J. Comput. 10 (1981) 328–337.
[17] J.A. Mesa, The conditional path center problem in tree graphs, unpublished paper presented to the EWGLA8 held in Lambrecht (Germany) in 1995.
[18] E. Minieka, Conditional centers and medians on a graph, Networks 10 (1980) 265–272.
[19] E. Minieka, The optimal location of a path or tree in a tree network, Networks 15 (1985) 309–321.
[20] E. Minieka, N.H. Patel, On finding the core of a tree with a specified length, J. Algorithms 4 (1983) 345–352.
[21] P.B. Mirchandani, A.R. Odoni, Locating new passenger facilities on a transportation network, Transport. Res. B 13B (1979) 113–122.
[22] C.A. Morgan, J.P. Slater, A linear algorithm for a core of a tree, J. Algorithms 1 (1980) 247–258.
[23] S. Peng, W. Lo, Efficient algorithms for finding a core of a tree with specified length, J. Algorithms 20 (1996) 445–458.
[24] S. Peng, A.B. Stephens, Y. Yesha, Algorithms for a core and k-tree core of a tree, J. Algorithms 15 (1993) 143–159.
[25] S. Sahni, General techniques for combinatorial approximations, Oper. Res. 25 (1977) 920–936.
[26] M. Sharir, P.K. Agarwal, Davenport–Schinzel sequences and their geometric applications, Cambridge University Press, 1995.
[27] A. Shioura, M. Shigeno, The tree center problems and the relationship with the bottleneck knapsack problems, Networks 29 (1997) 107–110.
[28] P.J. Slater, Locating central paths in a graph, Transport. Sci. 16 (1982) 1–18.

[29] A. Tamir, Fully polynomial approximation schemes for locating a tree-shaped facility: a generalization of the knapsack problem, Discrete Appl. Math. 87 (1998) 229–243.
[30] A. Tamir, T.J. Lowe, The generalized p-forest problem on a tree network, Networks 22 (1992) 217–230.
[31] A. Tamir, J. Puerto, D. Pérez-Brito, The centdian subtree on tree networks, Discrete Appl. Math. 118 (2002) 263–278.
[32] B.-F. Wang, Efficient parallel algorithms for optimally locating a path and a tree of a specified length in a weighted tree network, J. Algorithms 34 (2000) 90–108.
[33] B.-F. Wang, Finding a two-core of a tree in linear time, SIAM J. Discrete Math. 15 (2002) 193–210.
[34] B.-F. Wang, Private communication, June 2001.