Collection Depots Facility Location Problems in Trees

Robert Benkoczi
School of Computing, Queen's University,
Kingston (ON) K7L 3N6, Canada
rbenkocz@cs.queensu.ca,

Binay Bhattacharya
School of Computing Science, Simon Fraser University,
Burnaby (BC) V5A 1S6, Canada
binay@cs.sfu.ca,

Arie Tamir
Department of Statistics and Operations Research,
School of Mathematical Sciences, Tel Aviv University,
Ramat Aviv, Tel Aviv 69978, Israel
atamir@post.tau.ac.il

March 20, 2006

Abstract

We consider a generalization of the median and center facility location problem called the *collection depots facility location* (CDFL) problem. We are given a set of client locations and a set of collection depots, and we are required to find the placement for a certain number of facilities so that the cost of dispatching a vehicle from a facility, to a client, to a collection depot, and back, is optimized for all clients. The CDFL center problem minimizes the cost of the most expensive vehicle tour among all clients and the CDFL median problem minimizes the sum of the tour costs for all clients. We provide the first algorithms to solve the 1 and k median problems in trees with time complexities $O(n \log n)$ and $O(kn^3)$ respectively, where n is the number of vertices in the tree. In contrast, a restricted version of the k-median problem where clients are given lists of allowed collection depots, is NP-complete even for star graphs. We also give an optimal linear time algorithm to solve the discrete and continuous weighted 1-center problem, improving on the $O(n \log n)$ result of Tamir and Halman [10].
Keywords: Algorithms, dynamic programming, facility location, collection depots problem, trees.

1 Introduction

The collection depots facility location (CDFL) problem is an optimization problem with a special objective function. Two sets of points are given, a) clients and b) collection depots. The two sets need not be disjoint, *i.e.* a point can simultaneously be a client and a collection depot. A set of facilities provides service to all clients and this service requires that a vehicle originating at a facility travels to the client, then visits a collection depot, and finally returns to the same facility. The cost associated with this service is the weighted distance traveled by the vehicle visiting the client, *i.e.* the traveled distance multiplied by a positive value specific to the client. The goal is to compute the placement for k facilities so that a certain objective function is minimized. The objective function we consider here is (i) to minimize the cost of the most expensive tour (a k-minmax or k-center problem), or (ii) to minimize the total cost of all tours (a k-minsum or k-median problem). If every client location is also a collection depot, then we obtain an instance of the classic k-median or k-center problem. Since the classic k-median and k-center are NP-hard problems, naturally their CDFL versions are also NP-hard. In this paper we focus only on the CDFL location problems. We simply call these problems the k-median or k-center and we use the term "classic" k-median and k-center to refer to the usual location problems using the weighted distances for cost.

The CDFL location problem was recently proposed by Drezner and Wesolowsky [6] who motivated the research with several industrial applications such as septic tank cleaning service, garbage collection, or supply management. Their paper characterizes some properties of the optimal solution for the median problem on the line and in the plane with Euclidean and rectilinear distances. It also proposes a heuristic algorithm to solve the Euclidean distance version in the plane and presents an empirical study of its performance. In a subsequent paper [4], Berman, Drezner, and Wesolowsky considered the CDFL problem in general graphs and trees, but they only analyzed the optimal solution for the single facility median and center problems without proposing an algorithm to compute the optimal solution. This work was later extended by Berman and Huang [5] who studied the multi-facility location problems in general graphs, trees, and cycles. They established new properties for the optimal solution in trees and in cycles, and also proposed an algorithm for locating both collection depots and facilities in general graphs. Their method uses a Lagrangean relaxation algorithm embedded in a branch and bound framework. Recently, Tamir and Halman [10] studied the k-center problem in the plane, in graphs, in trees, and in paths. They proposed a more general formulation in which every client is given a set of collection depots that the client is allowed to use. In addition, they considered two extensions of the CDFL problem, the customer one way and the depot one way problem in which the cost of the return trip to the facility is ignored. Their paper contains a host of algorithmic results including constant factor approximation algorithms for k-center in general graphs, exact algorithms for 1-center in graphs and in the plane, and exact algorithms for k-center in trees and paths.

Center problems are classified based on where the facility can be placed and whether clients are weighted or not. If the facilities must reside at the vertices of the tree, we have an instance of the discrete center problem, otherwise the problem is called continuous (the facilities can be located on the edges of the tree). Tamir and Halman [10] show that a solution to the k-center CDFL problem in trees is obtained in $O(n^2 \log n)$ time for both discrete and continuous weighted problems. For the 1-center problem on a tree, they describe an algorithm with $O(n \log n)$ time complexity and a linear time algorithm for paths. We mention that the center problems considered by Tamir and Halman are more general, in that each client location has a list of allowed depots it can use. We refer to their version of the problem as the restricted median or center problem. We also note that for the median problem, the distinction between discrete and continuous versions is irrelevant because there is always an optimal solution to the continuous version consisting only of medians located at the vertices of the tree [5].

Our results can be summarized as follows. For the unrestricted median problem, we give an $O(n \log n)$ algorithm for locating one facility and a dynamic programming algorithm with time complexity $O(kn^3)$ for k facilities in trees. We also show that the restricted k-median problem on trees is NP-complete when the facility setup costs are not identical. For the unrestricted 1-center, we give an optimal linear-time algorithm for the weighted discrete and

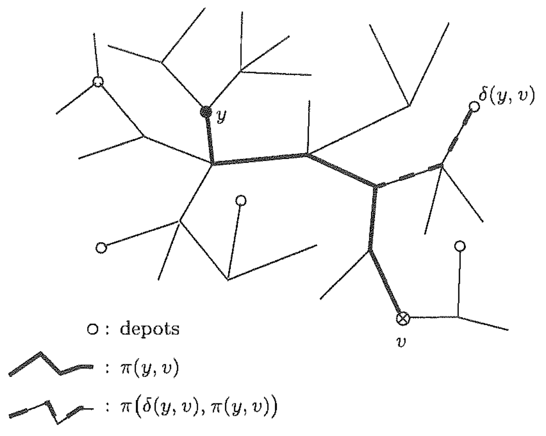

Figure 1: The trip to serve client v from facility y

continuous case, which is a generalization of the prune-and-search method used by Megiddo to solve the classic weighted 1-center problem in trees [7]. Our method does not apply for the restricted problem. We begin Section 2 by introducing the notation and proving properties that are crucial for the correctness of our algorithms. In Section 3 we discuss our algorithm for the single facility center problem. The algorithms for the median problems are given in sections 4.2 and 4.3.

2 Notation, Definitions, and Properties

Let $T = (C \cup D, E)$ be a tree where $|C| = n_C$, $|D| = n_D$, and $|C \cup D| = n$. Let $C = \{v_1, v_2, \ldots, v_{n_C}\}$ be the client vertices of T and let $D = \{\delta_1, \delta_2, \ldots, \delta_{n_D}\}$ be the depot vertices of T. Client and depot vertices may coincide. Each edge $e \in E$ has a positive length $l(e)$. We view the tree as a network in which an edge $e \in E$ corresponds to a closed interval $A(e)$ of length $l(e)$ so that we can uniquely identify its interior points by the distance to the endpoints of $A(e)$. We denote the set of all intervals of the edges of T by $A(T)$ (the set of arcs of T). A facility may be located at a point from set $A(T)$. Each client $v \in C$ is associated with weight $w(v) \geq 0$. If T' is a subtree of T, we denote by $C(T')$ and $D(T')$ the client respectively depot set of T'. For any pair of points x and y from $A(T)$, let $\pi(x, y)$ be the unique path between x and y, and let $d(x, y)$ be the network distance between x and y defined as the length of path $\pi(x, y)$. We extend this notation to include the network distance between a point x and a path $P \subseteq A(T)$, defined as

$$d(x, P) = \min_{v \in P} d(x, v).$$

Similarly, we let $\pi(x, P)$ be the path between point x and path P. For simplicity, we use the term *tree* even when we refer to the points $A(T)$ from the network defined by tree T.

The restricted version of the collection depots problem proposed by Tamir and Halman [10] specifies that every client v_i is associated a set $X(v_i)$ of depots that are allowed for that client. If $X(v_i) = D$ for all clients v_i, then we have an instance of the unrestricted CDFL problem. Unless we specify otherwise, all location problems discussed in this paper are unrestricted. Let y be a point in $A(T)$. We denote the weighted trip distance, also called the trip cost, from facility y to client v_i by

$$r(y, v_i) = w(v_i)\Big(d(y, v_i) + \min_{\delta \in X(v_i)} \{d(v_i, \delta) + d(\delta, y)\}\Big).$$

Figure 1 shows a typical example of a route from facility y to client v. The optimal depot location for the route determined by y and v is denoted $\delta(y, v)$ and is not necessarily the closest depot to the client or to the facility, but it depends on the facility-client pair.

Problem 1 (Unrestricted CDFL facility location). *For any finite subset of points $F \subseteq A(T)$ in the tree network, we define the following two objective functions which correspond to the median and center location problems respectively.*

$$\begin{aligned} m(F) &= \sum_{i=1}^{n_C} \min_{y \in F} r(y, v_i) \\ c(F) &= \max_{1 \le i \le n_C} \min_{y \in F} r(y, v_i) \end{aligned}$$

Input:

- *A tree $T = (V, E)$, with the set of clients C, set of depots D, and edge length function l.*
- *A non-negative client weight function $w : C \to \mathbb{R}_+$.*
- *A non-negative function for the cost of opening facilities at the vertices of the tree, $f : V \to \mathbb{R}_+$. For any subset of vertices V', we write $f(V') = \sum_{y \in V'} f(y)$.*
- *A positive integer parameter k.*

Output: *A set F_k of cardinality k of facilities that optimizes the following objective functions.*
k-median: *The facilities are opened on the vertices, and the total service cost is minimized,*

$$F_k \subset V, \quad m(F_k) + f(F_k) = \min_{\substack{Y \subset V \\ |Y| = k}} \big(m(Y) + f(Y)\big).$$

Discrete k-center: *The facilities are opened on the vertices, and the maximum service cost is minimized,*

$$F_k \subset V, \quad c(F_k) = \min_{\substack{Y \subset V \\ |Y| = k}} \big(c(Y)\big).$$

Continuous k-center: *The facilities can be opened on the vertices and the edges, and the maximum service cost is minimized,*

$$F_k \subset A(T), \quad c(F_k) = \min_{\substack{Y \subseteq A(T) \\ |Y| = k}} c(Y).$$

For center problems, there is no cost of opening facilities.

If the optimal depot used in the trip from y to v_i is determined, we can write

$$r(y, v_i) = w(v_i)\Big(d(y, v_i) + d\big(v_i, \delta(y, v_i)\big) + d\big(\delta(y, v_i), y\big)\Big).$$

If we analyze the trip from y to v_i and keep in mind that T is a tree network, we notice that the trip from y to v_i can be partitioned into two parts. One part involves the path from y to

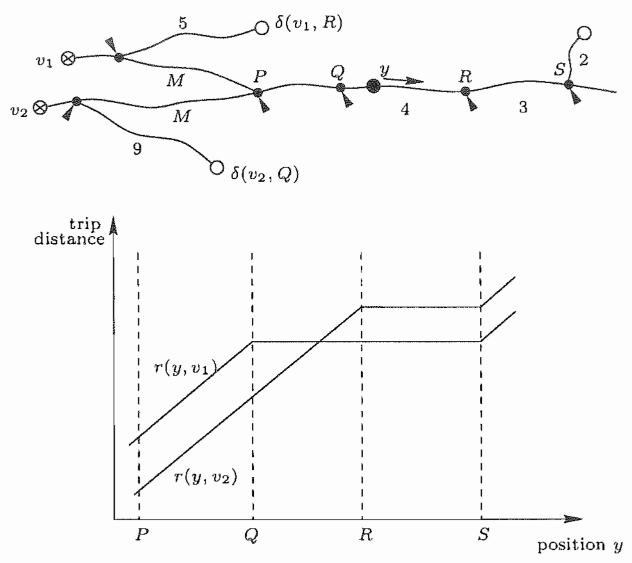

Figure 2: Here, distance M is assumed to be sufficiently large so that depots $\delta(v_1, R)$ and $\delta(v_2, Q)$ do not influence each other; somewhere between Q and R, the dominating client changes

v_i, denoted by $\pi(y, v_i)$, and the other involves the path from $\delta(y, v_i)$ to $\pi(y, v_i)$ (see Fig. 1). Hence

$$r(y, v_i) = 2w(v_i)\Big(d(y, v_i) + d\big(\delta(y, v_i), \pi(y, v_i)\big)\Big). \tag{1}$$

The unweighted trip distance $\frac{r(y,v_i)}{w(v_i)}$ has certain properties that are different from the usual tree distance $d(y, v_i)$. We point out one difference that causes difficulties in our algorithms for both median and center problems but first, we mention an intuitive result proved by Tamir and Halman. They show ([10] Lemma 4.3) that the trip distance to any client is monotone non-decreasing as the facility moves on a simple path away from the client. This property is used in their 1-center algorithm on trees. We also use it here in the design of our improved linear-time algorithm. Unfortunately, if we consider a set of clients and a facility y that moves on a simple path, away from all of them, the client with the largest (smallest) unweighted trip distance from y can change as y moves. This does not happen if we use the unweighted tree distance. An example is given in Fig. 2.

3 The Weighted 1 Center CDFL Problem

In this section we present a linear time algorithm to solve both the discrete and continuous unrestricted weighted 1-center CDFL problems in trees. For the restricted problem, the best known result is from Tamir and Halman [10] with complexity $O((n + K) \log n)$ where $K = \sum_{i=1}^{n_C} |X(v_i)|$ is the total size of allowed depots which can be $O(n^2)$. We generalize the iterative approach of Megiddo [7] used to solve the classic 1-center problem to the CDFL problem. More precisely, we modify the $O(n \log n)$ iterative algorithm of Tamir and Halman [10] so that we eliminate from the tree a constant fraction of vertices at every iteration. We assume the tree to be rooted so that we have pairs of vertices in a child-parent or ancestor-descendant relationship. If the tree is not rooted, we choose an arbitrary vertex as root.

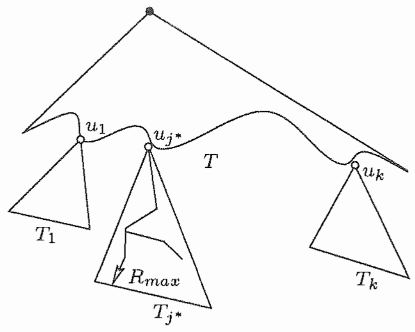

Figure 3: Determining whether the optimal center location belongs to a forest of subtrees of T

Let $T(i)$ be the tree consisting of the vertices not eliminated at iteration $i-1$ or at earlier iterations. Let c be the centroid of $T(i)$. Iteration i consists of computing all weighted trip distances from c as a facility to all clients in $T(i)$ in time $O(|T(i)|)$ (see [10]) and determining the client x_{max} with the largest weighted trip distance. We denote by $S_1(c)$ the component rooted at c that contains x_{max} and by $S_2(c)$ the component, also rooted at c, that doesn't contain x_{max} (see Fig. 4). Both $S_1(c)$ and $S_2(c)$ have more than $\lfloor\frac{1}{3}|T(i)|\rfloor$ vertices each. Because of the monotonicity of the unweighted trip distance, we know that the optimal center must reside in $S_1(c)$ and therefore we can recursively select the centroid of $S_1(c)$ as the candidate facility for the next iteration. To achieve a running time linear in $n = |T|$, we will identify a constant fraction of clients from $S_2(c)$ (and thus a constant fraction of the total number of vertices used at iteration i) that cannot determine the position of the optimal center because they are dominated by other clients from $T(i)$. If x_{max} is not unique and an equidistant vertex exists in $S_2(c)$, then c is the optimal solution for both the discrete and continuous 1-center problems.

Our approach requires an answer to the following query in time linear in the size of the tree. Let T_j for $1 \leq j \leq k$ be a forest of k subtrees, so that u_j is the root of T_j and T_j consists of all the descendants of u_j. Does the optimal center y_{opt} belong to the union of these subtrees or not? To answer this query, we use the idea of Megiddo [7] (see Fig. 3). From u_j, we can compute the weighted trip distance to all clients in T_j in time proportional to the size of T_j. Let R_j be the maximum weighted trip distance over the clients in T_j and let $R_{max} = R_{j^*} = \max_{1\leq j\leq k} R_j$. It is possible to show using arguments similar to those of Megiddo in [7] that, if $y_{opt} \in \bigcup_{j=1}^{k} T_j$, then $y_{opt} \in T_{j^*}$. Indeed, if y_{opt} is in T_j and $j \neq j^*$, then no client from T_j can determine the position of y_{opt} because there is at least a client in T_{j^*} with a dominating weighted trip distance. Consequently, we can move y_{opt} away from T_j without increasing the cost of the 1-center solution. It follows that, to determine if the optimal center is in $\bigcup_{j=1}^{k} T_j$, it suffices to compute the trip distances from u_{j^*} to all the clients in time linear in the size of the tree. If the largest trip distance from u_{j^*} is to a client in T_{j^*}, we conclude that $y_{opt} \in \bigcup_{j=1}^{k} T_j$, otherwise y_{opt} is not in the union of subtrees.

We now describe the process of pruning vertices from the tree. As in the previous section, the optimal center must lie in component $S_1(c)$ where c is the centroid found at an iteration. We will identify a constant fraction of the client vertices in $S_2(c)$ that will not determine the position of the optimal center. These client vertices will be eliminated from the tree. Moreover, the tree structure of component $S_2(c)$ is not needed anymore and can be replaced by a list of client vertices $v \in S_2(c)$. The list contains, for every client v, the distance to the centroid $d(v, c)$ and the depot distance from the closest restricted depot $\delta'(c, v)$ located

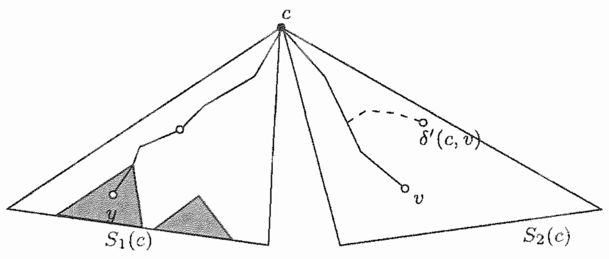

Figure 4: The case when the optimal facility is in component $S_1(c)$

in $S_2(c)$ to the path $\pi(c,v)$. We call such a depot restricted and we use notation $\delta'(c,v)$ if this depot is the closest depot located in some given subtree of the tree. If the given subtree (for example $S_2(c)$) contains no depot, then $\delta'(c,v)$ is not defined and the corresponding depot distance is considered infinitely large. Using the list of centroid and restricted depot distances for clients in $S_2(c)$, the length of the trip originating at any vertex in the other component $S_1(c)$ to any client in $S_2(c)$ can be easily computed. In Fig. 4, if y is the origin and v is the client, the optimal depot used is either $\delta'(c,v)$ or another depot from $S_1(c)$ depending on how far this other depot is from path $\pi(y,c)$. It is not difficult to modify the trip distance computation from a fixed source described in [10] to handle the lists of clients associated with certain vertices in the tree structure.

Let δ'_{med} be the median value of depot distances $d\big(\delta'(c,v),\pi(c,v)\big)$ from the list associated with component $S_2(c)$, and denote by K^+ the clients from $S_2(c)$ with depot distance greater or equal than δ'_{med} and K^- the clients with depot distance smaller or equal to δ'_{med}. Thus both K^+ and K^- have cardinality at least $\lfloor\frac{|S_2(c)|}{2}\rfloor$. If component $S_2(c)$ contains no depot, then we set $K^+ = S_2(c)$ and $K^- = \emptyset$. We focus now on the vertices from the other component, $S_1(c)$. If we look at the path from a vertex $y \in S_1(c)$ to c, there exists a depot in $S_1(c)$ (a restricted depot!) that is closest to the path. For some vertices in $S_1(c)$ this restricted depot distance could be smaller or equal than δ'_{med}. These vertices form a forest of subtrees of $S_1(c)$ that we denote $T_1,\ldots,T_k$ (the shaded subtrees in Fig. 4).

We know how to determine whether the optimal center y_{opt} lies in $\bigcup_{j=1}^k T_j$ or not in time linear in $|T(i)|$. First, assume $y_{opt} \in \bigcup_{j=1}^k T_j$ and therefore the depot distance from y_{opt} to centroid c is smaller than δ'_{med}. This means that all clients from set K^+ will use the same depot from component $S_1(c)$ if served by y_{opt}. We arbitrarily form pairs of clients from set K^+. For every such pair (u,v), we compute a value denoted t_{uv} that represents the additive unweighted cost needed to make the weighted trip distances to u and v equal. More precisely, t_{uv} is such that

$$w(v)\cdot\Big(t_{uv}+d(c,v)\Big) = w(u)\cdot\Big(t_{uv}+d(c,u)\Big).$$

Let t_{med} be the median value of the t_{uv} computed above. We are now interested in the vertices $y \in S_1(c)$ for which the unweighted trip distance from y to c is greater than t_{med}. It is not difficult to observe that the set of all y vertices form again a forest of disjoint rooted subtrees of $S_1(c)$, and thus we can determine again if the optimal center y_{opt} belongs to this forest or not. In other words, we can determine if $d(y_{opt},c)+d\Big(\delta(y_{opt},c),\pi(y_{opt},c)\Big)$ is greater than t_{med} or not. If it is greater, then from the pairs of clients in $S_2(c)$ with $t_{uv} \le t_{med}$, the weighted trip distance from y_{opt} to one of the clients will always dominate the weighted trip distance to the other. Thus we can eliminate the client with the dominated weighted trip distance. If the trip portion from y_{opt} to the centroid is smaller than t_{med}, then we focus on the pairs of clients with $t_{uv} \geq t_{med}$. Here too, the weighted trip distance to one of the clients will dominate the trip distance to the other client in the pair. In each case, one quarter of the vertices from set K^+ is pruned, *i.e.* at least $\lfloor \frac{1}{8}|S_2(c)| \rfloor$.

In the second case, $y_{opt} \in S_1(c) \setminus \bigcup_{j=1}^{k} T_j$. This means that any client in $S_2(c)$ from set K^- will use its own depot from component $S_2(c)$ in the trip originating at y_{opt}. We again arbitrarily pair up the clients from K^-. For every pair (u, v), we compute the value t'_{uv} that represents the distance to centroid c from a vertex y in $S_1(c)$ required so that the weighted trip distances from y to both u and v are equal. More precisely, t'_{uv} satisfies

$$w(v) \cdot \Big(t'_{uv} + d(c,v) + d(\delta(c,v), \pi(c,v))\Big) = w(u) \cdot \Big(t'_{uv} + d(c,u) + d(\delta(c,u), \pi(c,u))\Big).$$

The difference from the first case is that the depots used are in both cases chosen from $S_2(c)$ and do not depend on the choice of facility y. Let t'_{med} be the median value of t'_{uv} above. The vertices y from $S_1(c)$ that are further than t'_{med} from the root form a forest of subtrees. We can again determine if y_{opt} lies in this forest or not. Exactly as in the previous paragraph, we identify one vertex from at least half of the pairs (u, v) that can be pruned.

As a final technical detail, we point out that component $S_1(c)$ in which we search for the optimal center may contain only partial structure information. Some rooted subtrees of $S_1(c)$ are represented only by lists of clients and restricted depot distances. However, it is not a problem to decide if any forest of subtrees of $S_1(c)$ contains the optimal center because we already know that the subtrees represented by these lists do not contain y_{opt}.

Analysis: At every iteration, computing the weighted trip distances to all client vertices and determining where the optimal center lies is performed in time linear in the size of the tree. Pruning also eliminates a quarter of the vertices of set K^+ or K^-. Each of these sets contains at least half of the number of clients from component $S_2(c)$. This component also contains at least one third of the vertices in tree $T(i)$ used at iteration i. Therefore, iteration i prunes at least $\frac{1}{24}|T(i)|$ vertices and therefore the algorithm has time complexity $O(n)$, where $n = |T|$.

Computing the optimal discrete center: The iterative process described above stops when either (i) c is equally distant to at least two dominating vertices from $S_1(c)$ and $S_2(c)$, in which case c is the optimal center, or (ii) $|T(i)| < 24$, in which case we find the optimal center in constant time by complete enumeration.

3.1 The Continuous 1-center CDFL Problem

To solve the continuous 1-center problem, we first compute the discrete optimal 1-center using the procedure described above. If the algorithm stops because c is equidistant to two dominating vertices from both $S_1(c)$ and $S_2(c)$, then c is the solution for the continuous problem too (case i). Otherwise, $|T(i)| < 24$ and we compute the optimal continuous center by enumeration, in constant time.

4 The Median Problem

In this section, we describe our solution for the 1-median and k-median CDFL problems. We consider the unrestricted median problems. We show that k-median can be solved in polynomial time with dynamic programming and that the 1-median problem is solvable in $O(n \log n)$ time and linear space even when the cost of opening facilities is arbitrary. In

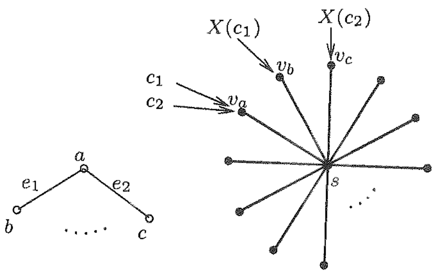

Figure 5: NP-completeness of restricted k-median, reduction from the vertex cover problem. The example assumes $a \prec b \prec c$.

contrast, the restricted k-median problem with non-uniform costs for opening facilities is NP-complete.

Theorem 4.1. *The restricted k-median problem with non-uniform costs for opening facilities is NP-complete even on star graphs and even when the list $X(v_i)$ of restricted depots for client v_i is a singleton for all v_i.*

Proof. The restricted k-median problem is in NP because the cost of any solution can be computed in polynomial time. To prove the NP-hardness, we use a reduction to the vertex cover problem. Given a graph $G = (V, E)$, we are asked if a vertex cover of size k exists in G.

First we fix an arbitrary permutation of the vertices in G which is represented by the binary relation $\prec$. In other words $a \prec b$ if and only if a precedes b in the permutation. Then we construct a star graph $G' = (V', E')$ with the central vertex s and a vertex v_a of degree one for each vertex $a \in V$. We let $f(s) \to \infty$ and $f(v_a) = 0$ for all $v_a \in V' \setminus \{s\}$. Let $e_i = (u, v)$ be an edge in graph G and assume without loss of generality that $u \prec v$. With e_i, we associate a client c_i placed at vertex u in the star graph and we set $X(c_i) = \{v\}$. We do this for all edges in the graph G. Clearly, a vertex of star graph G' may contain more than one client. We also set the edge lengths and vertex weights of the star graph to 1. Then, a vertex cover of size k exists in G if and only if the cost of the restricted k-median in G' is not more than $2m$, where $m = |E|$. □

Our algorithm for the 1 facility minsum problem uses a tree decomposition called the *spine decomposition* that has been previously used in the context of median problems. For completeness, we give a brief description of this structure in the following section.

4.1 The Spine Decomposition (SD)

The spine decomposition (SD) is a tree decomposition designed and presented in some of our earlier work, [3, 1]. The SD uses a path to direct the partition of the input tree. This makes it suitable for computation tasks involving components of the tree that interact with one another.

In Fig. 6, let T be a binary tree rooted at a vertex r_T. If T is not binary, we can make it binary as in [9]. We select a path from r_T to a leaf in T such that the next vertex in this path always follows the child with the most number of leaves hanging from it. Formally, if $v_0 = r_T$, v_1, ... v_k are the vertices on the path, if $p(v)$ denotes the parent of v, and if $N_l(v)$ denotes the number of leaves that have v as ancestor, then we always have $N_l(v_{i+1}) \geq N_l(u_i)$ where u_i is the other child of vertex v_i. We call path $v_0, v_1, \ldots v_k$ a spine and use notation $\pi(v_0, v_k)$. If

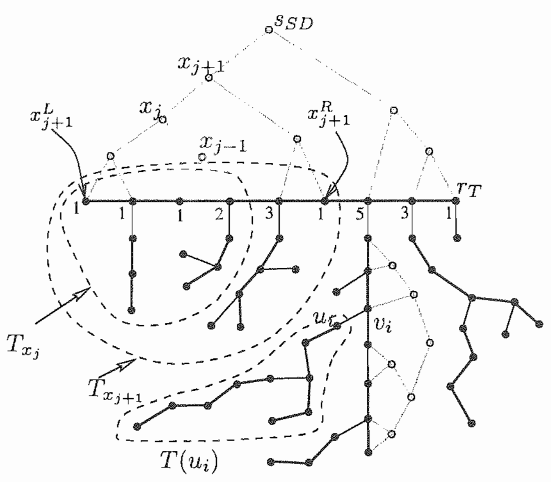

Figure 6: A typical spine decomposition; spines are shown in thick lines, search trees as thin lines and components are outlined by dashed lines; the numbers beside spine vertices at the top-most spine give the number of leaves of T for the corresponding SD component

we remove the spine from T we obtain a set of at most k disconnected components which are each recursively decomposed. Let $T(u_i)$ be one of these components rooted at u_i where u_i is adjacent to spine vertex v_i but is not itself a spine vertex. We call $T(u_i)$ an SD component.

Let $\lambda(v_i)$ denote the number of leaves of $T(u_i)$. We construct a binary search tree with vertices v_i as leaves and we associate it with the spine. Thus the leaves of the search tree and the spine vertices of $\pi(v_0, v_k)$ are the same (or one can consider a one to one mapping between them). The root of the search tree is linked to the spine vertex of the parent spine, *i.e.* the root of the search tree for the spine of $T(u_i)$ is linked to v_i. In this way, different search trees are connected through one or more spine edges, and we can talk about a unique *super*-path that exists between any two search tree nodes that belong to different search trees. Search trees are balanced by weight $\lambda(v_i)$ associated with leaf v_i such that components with many leaves (and thus many recursive spines) are closer to the root of the search tree. We denote the path between any two search tree nodes x and y by $\sigma(x, y)$.

The SD components that are descendent (in a search tree) of some SD node x form a subtree of T denoted T_x and called subtree of x. Given x, we identify two spine vertices (and thus two vertices of the original tree) with x. The spine vertex denoted x^R (and x^L respectively), is the spine vertex that (1) has x as ancestor in the search tree and (2) is the closest (or furthest respectively) to r_T if we consider the tree distance in the original tree T. Two important properties of the SD are mentioned here without proof because of space constraints. Full proofs are provided in [1].

Theorem 4.2. *The length (number of edges) of any super-path $\sigma(x, y)$ in the SD is $O(\log n)$ where n is the number of vertices of the input tree T.*

Theorem 4.3. *The construction algorithm for the SD has time complexity $O(n)$. The storage space complexity of the data structure for the SD is also $O(n)$.*

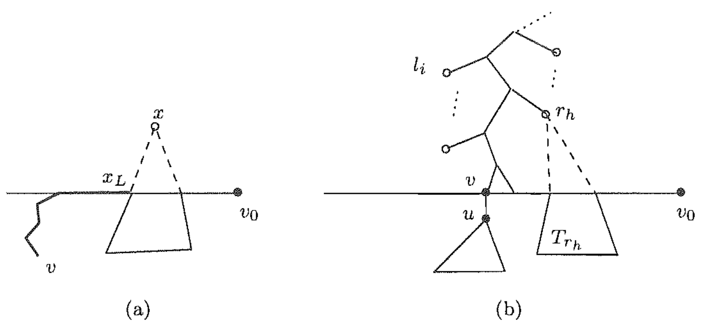

Figure 7: Computing the cost of the 1-median when v is the facility

4.2 The 1-median Problem

The general idea of the 1-median algorithm follows a pattern similar to the work of Rosenthal and Pino [8] who studied the location of one facility in trees, with several objective functions, in linear time. For this problem, the same framework seems not to lead to linear time algorithms very easily because of the peculiarities of our distance function. We propose here an algorithm with running time $O(n \log n)$ for locating one median.

From the paper of Berman and Huang [5], we know that the optimal solution must be a vertex of the input tree, therefore we simply compute the 1-median cost with the facility at each vertex in the tree and select the one for which the value computed is the smallest. The obvious algorithm to compute the cost for a given facility uses linear time for processing which leads to a quadratic 1-median algorithm. In the following paragraphs, we show that using the spine decomposition [3, 1] and pre-processing, we can compute the cost in logarithmic time per candidate median.

The main idea is as follows. We can compute, in $O(n \log n)$ time, the cost of the 1-median in subtree T_x if x^L or x^R is the median. With x^R as the median, we need to determine the contribution of the remaining vertices from $T \setminus T_x$ in order to estimate the cost in the whole tree. Because of the properties of the spine decomposition, there are $O(\log n)$ subtrees whose costs need to be accounted for. However, the depot distance from x^R to the root of those subtrees influences the cost of the subtrees. This influence can be calculated if we sort the vertices z in each subtree by the restricted depot distance from the root of the subtree to z and if we precompute prefix sums of weighted distances.

Assume that at every node x of the SD we have the following information available (Fig. 7).

(a) The clients of T_x are sorted in decreasing order of the distance from the path between the client and x_L, and the closest depot to the path. The sequence is z_1, z_2, ... $z_{|C(T_x)|}$ such that

$$d(\delta(x_L, z_j), \pi(x_L, z_j)) \geq d(\delta(x_L, z_{j'}), \pi(x_L, z_{j'}))$$

for any $j' > j$. Note that $\delta(x_L, z_j)$ is the closest depot to the path from the set of all depots. It can be computed efficiently using the approach in [10, 1].

(b) The weighted sum for the depot distance to path $\pi(z_j, x_L)$ for all clients starting with z_j to the last one in the order described above. We use subscript "L" because we compute another value $P_R(x, j)$ which returns the same weighted sum but using the ordering relative to paths $\pi(z_j, x_R)$.

$$P_L(x,j) = \sum_{i=j}^{|C(T_x)|} w(z_i) \cdot d\big(\delta(x_L, z_i), \pi(x_L, z_i)\big), \quad 1 \le j \le |C(T_x)|.$$

(c) The sum of the weights of the clients in the order described at (a) up to vertex z_j.

$$Q_L(x,j) = \sum_{i=1}^{j} w(z_i), \quad 1 \le j \le |C(T_x)|.$$

(d) The cost of all clients in T_x as if served by a facility at x_L but without adding the depot distance. To obtain the trip distance, one needs to add this value to the depot distance weighted sum.

$$M_L(x) = \sum_{i=1}^{|C(T_x)|} w(z_i) \cdot d(x_L, z_i).$$

A similar ordering of the clients in T_x is computed relative to the depot distance to the path from the client to x_R, $d\big(\delta(z_j, x_R), \pi(z_j, x_R)\big)$. Then, we define $P_R(x,j)$, $Q_R(x,j)$, and $M_R(x)$ exactly in the same way using the new ordering.

To compute the contribution of clients in T_x if served by some tree vertex v (Figure 7 (a)), we simply have to know the depot distance to path $\pi(v, x_L)$. Let this distance be d_{new},

$$d_{new} = d\big(\delta(v, x_L), \pi(v, x_L)\big).$$

Let j be the largest index in the ordering $z_1, \dots z_{|C(T_x)|}$ of the client vertices in T_x relative to x_L for which the depot distance is larger than d_{new}, in other words, for which

$$d\big(\delta(z_j, x_L), \pi(z_j, x_L)\big) > d_{new}.$$

Then, the contribution of clients in T_x served by v is

$$2 \cdot \big(M_L(x) + w(T_x) \cdot d(v, x_L) + Q_L(x,j) \cdot d_{new} + P_L(x, j+1)\big). \qquad (2)$$

Indeed, the first two terms represent the total cost for the trip between the clients and the facility and the last two the total cost for the trip between the optimal depot and the client-facility path. Note that we use d_{new} as depot distance for all clients for which the depot distance for the trip inside T_x is larger than d_{new}. Of course, if v is towards the root from x, we use the values P_R, Q_R and M_R in a similar way.

Now, we have all the ingredients needed to compute the 1-median cost when some vertex $v \in T$ is the facility. Consider Figure 7 (b) where $v \in T$ is the facility for which we need the cost. Let $r_0, r_1, \dots r_a$ and $l_0, l_1, \dots l_b$ be the SD nodes adjacent to path $\sigma(v, s_{SD})$. At each of these SD nodes including v, we use (2) to evaluate the contribution of the client vertices in the respective components, and we report the total as the result.

Observe that the evaluation of (2) is done in constant time once value j is determined. If we use binary search with d_{new} over the ordering of clients, we spend $O(\log n)$ time at each SD node, and thus $O(\log^2 n)$ time for each vertex. This gives an algorithm with $O(n \log^2 n)$ running time. However, we can replace the binary search step with sequential search if we have access to d_{new} in sorted order. Let y be the SD node sibling of x. Node x is used in the computation from (2) only when $v \in T_y$. But at y, we already have the sorted list of all vertices (we will compute the sorted list for all vertices and not only for the clients) relative to their depot distance to the path to either y_L or y_R. We then evaluate (2) for v in this order. We can also reduce the storage space of this algorithm to $O(n)$ from the obvious bound of $O(n \log n)$ if we discard the lists of values that are not needed in the algorithm. The algorithm is sketched below.

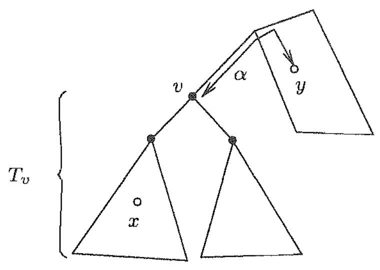

Figure 8: Cost function for the k-median problem

- Compute the SD and any information required for maintaining the sorted lists of vertices at any node.
- Traverse the SD bottom up; at every node x with children y and t do:
  1: Sort the clients in T_x by depot distance; let $Z(x)$ be the sorted list of clients. Compute the lists of values P, Q, and M for $Z(x)$. Store everything at x.
  2: Traverse $Z(y)$ and generate queries in t (*i.e.* evaluate the contribution of T_t if the median is at a vertex in T_y using (2)). Answer the queries by sequential search in the list at t.
  3: Store the result incrementally in an array indexed by the tree vertex corresponding to the query.
  4: Repeat steps 2 and 3 with the roles for y and t interchanged.
  5: Discard the lists stored at y and t.
- Traverse the array indexed by tree vertices and output the entry with smallest value.

We can state the following result.

Theorem 4.4. *The* 1*-median collection depots problem in trees can be solved in* $O(n \log n)$ *time and* $O(n)$ *space.*

4.3 The Unrestricted k-median Problem

In this section we show that the unrestricted k-median problem can be solved in polynomial time for trees. We use dynamic programming like in the classic k-median problem. We assume that T is a binary rooted tree, otherwise we can root it at an arbitrary vertex and make it binary by adding a linear number of vertices as in Tamir's approach for the classic problem [9]. The cost functions we propose are more complex because they have two parameters that depend on the choice of facilities, and not only on one as in the classic version. The cost functions are associated with subtrees of the given rooted tree.

Let v be the root of subtree T_v and p the number of facilities chosen in T_v (see Fig. 8). For $x, y \in V$, we define the function $G(T_v, p, x, y)$ to be the minimum sum of trip costs for the clients in T_v plus the facility opening costs for the facilities in T_v if p facilities are chosen in T_v optimally, x is a facility that serves client v, and the nearest facility in tree distance to v is y. If $x \in T_v$, then x counts towards the number p of facilities chosen in T_v and the opening cost $f(x)$ for this facility is added to the value of the cost function. Similarly, if the closest facility y is in T_v and is different from x, it too counts towards the number p and the opening cost $f(y)$ is added to the value of the function.

The dynamic programming algorithm computes the cost functions for all the rooted subtrees T_v of tree T and all possible choices of parameters, bottom up. If r_T denotes the root of T, then the optimal unrestricted k-median solution is retrieved from $\min_{x,y \in V} G(T_{r_T}, k, x, y)$. Before we analyze the recursive computation of cost function G, we prove its correctness.

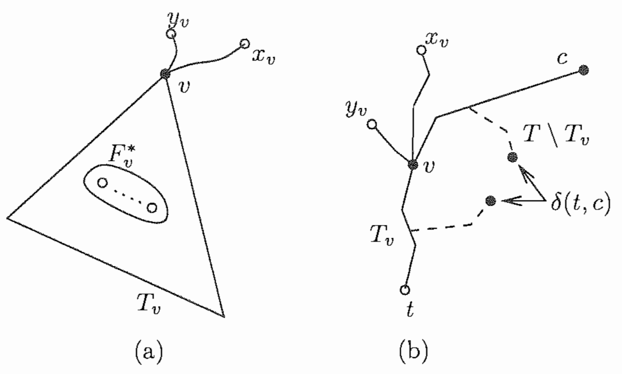

Figure 9: Correctness of cost function G

Theorem 4.5 (Principle of optimality). *Let F^* be an optimal set of k facilities for the k-median problem and let $v \in V$ be an arbitrary vertex with T_v as its rooted subtree. Denote by x_v the facility from F^* that serves v and by $y_v \in F^*$ the closest facility to v. Then, for $p = |F^* \cap T_v|$, the contribution of clients and facilities in T_v in the optimal solution F^* is equal to $G(T_v, p, x_v, y_v)$.*

Proof. Let $F_v^* = F^* \cap T_v$ and let $p = |F_v^*|$ (see Fig. 9). To prove the result it is sufficient to show that for each client $c \in T \setminus T_v$ ($c \in T_v$), and for each facility $t \in F_v^*$ ($t \notin F_v^*$), we have $r(t,c) \geq \min\{r(y_v,c), r(x_v,c)\}$. This will imply that in the optimal solution F^*, a client $c \in T \setminus T_v$ is served by a facility in $(F^* \setminus F_v^*) \cup \{x_v, y_v\}$, and a client $c \in T_v$ is served by a facility in $F_v^* \cup \{x_v, y_v\}$. We consider only the case where $c \in T \setminus T_v$, since, due to symmetry, the proof for the case where $c \in T_v$ is basically identical.

Let $t \in F_v^*$ be a facility serving a client $c \in T \setminus T_v$ (Fig. 9-b). Consequently, v is on the path from t to c. Suppose that $\delta(t,c)$ is in T_v. Then,

$$r(t,c) = w(c) \cdot \Big(\frac{r(t,v)}{w(c)} + 2d(v,c)\Big) \geq w(c) \cdot \Big(\frac{r(x_v,v)}{w(c)} + 2d(v,c)\Big) \geq r(x_v,c).$$

Suppose that $\delta(t,c)$ is in $T \setminus T_v$. Then,

$$r(t,c) = 2w(c) \cdot \Big(d(t,v) + d(v,c) + d\big(\delta(t,c), \pi(v,c)\big)\Big) \geq$$
$$\geq 2w(c) \cdot \Big(d(y_v,v) + d(v,c) + d\big(\delta(t,c), \pi(v,c)\big)\Big) \geq r(y_v,c).$$

□

4.3.1 Computation of the cost function:

From the proof of the previous theorem we notice that the argument y of the cost function $G(T_v, p, x, y)$, which represents the nearest facility to v, can potentially serve clients from the other side even though v is served by x. More precisely, if $y \in T_v$, then y might serve clients from $T \setminus T_v$. This could reduce the overall cost even if the cost incurred strictly in T_v returned by the cost function $G(T_v, p, x, y)$ is higher than when y is not forced to be a facility. When $y \in T \setminus T_v$, y could serve clients from T_v. We call the case when y serves clients beyond vertex v, *cross-service*.

There is an important observation regarding a facility y which is responsible for cross-service. This observation follows directly from the proof of Theorem 4.5. Let v be an arbitrary

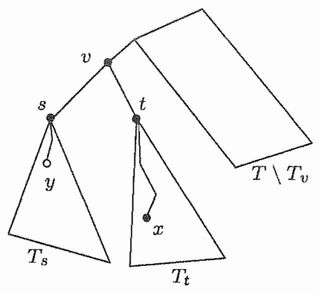

Figure 10: Vertex y and the three components adjacent to v

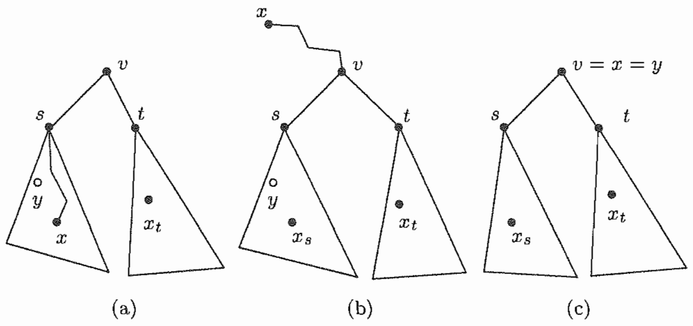

Figure 11: Recursive computation of cost function $G(T_v, p, x, y)$

vertex of T as in the proof of the theorem, and let s and t be its two children. Vertex v is adjacent to three components, T_s, T_t, and $T \setminus T_v$ (Fig. 10). Assume without loss of generality that $y \in T_s$. Then, either y provides cross-service to both T_t and $T \setminus T_v$, or only to one of them, or to neither one of them. In any case, we cannot have two distinct facilities y_1, y_2 with $d(y_1, v) \neq d(y_2, v)$ that provide cross-service. If y does not provide service to at least one client from T_t of $T \setminus T_v$ then y is called redundant for that component. We will compute the cost function $G(T_v, p, x, y)$ even for values of y that are redundant for T_v.

We now describe the computation of $G(T_v, p, x, y)$. We identify three base cases (Fig. 11). In each of these base cases, several sub-cases are possible depending on the location of vertex y.

A) $x \in T_v$, $x \neq v$. Without loss of generality, assume $x \in T_s$ (Fig. 11-a). If $x \in T_s$, then we can use the same argument as in the proof of Theorem 4.5 to show that x must also serve vertex s. Consequently, the contribution of clients in T_s is returned by $G(T_s, i, x, y)$. For the other subtree T_t, it is possible that other medians from within T_t serve the root of t. Therefore, to calculate the contribution of T_t, we need to choose the best cost function computed for T_t among all choices of $x_t \in T_t$.

We consider four sub-cases that depend on the location of y. We assume $y \neq v$ since otherwise $x = y = v$.

1) $y \in T_s$ and $y \neq x$.

$$G(T_v,p,x,y) = r(x,v) + \min_{2 \le i \le p} \Big\{ G(T_s,i,x,y) + \\ + \min\Big\{ G(T_t,p-i,x,y),\ G(T_t,p-i,y,y),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t,p-i,x_t,y) \Big\}\Big\}. \quad (3)$$

2) $y = x$. The recurrence relation follows the same pattern,

$$G(T_v,p,x,x) = r(x,v) + \min_{1 \le i \le p} \Big\{ G(T_s,i,x,x) + \\ + \min\Big\{ G(T_t,p-i,x,x),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t,p-i,x_t,x) \Big\}\Big\}. \quad (4)$$

3) $y \in T \setminus T_v$.

$$G(T_v,p,x,y) = r(x,v) + \min_{1 \le i \le p} \Big\{ G(T_s,i,x,y) + \\ + \min\Big\{ G(T_t,p-i,x,y),\ G(T_t,p-i,y,y),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t,p-i,x_t,y) \Big\}\Big\}. \quad (5)$$

4) $y \in T_t$.

$$G(T_v,p,x,y) = r(x,v) + \min_{1 \le i \le p-1} \Big\{ G(T_s,i,x,y) + \\ + \min\Big\{ G(T_t,p-i,x,y),\ G(T_t,p-i,y,y),\ \min_{\substack{x_t \in T_t \\ i \le p-2}} G(T_t,p-i,x_t,y) \Big\}\Big\}. \quad (6)$$

B) $x \in T \setminus T_v$ (Fig. 11-b). We have a similar situation except that besides t, the client s could also be served by some facility located in T_s and thus we need to select the optimal cost function for T_s as well.
Three sub-cases are identified depending on the choice for y as long as $y \neq v$.
1) $y \in T_v$. Without loss of generality we assume $y \in T_s$.

$$G(T_v,p,x,y) = r(x,v) + \\ + \min_{1 \le i \le p} \Big\{ \min\Big\{ G(T_s,i,x,y),\ G(T_s,i,y,y),\ \min_{\substack{x_s \in T_s \\ x_s \neq y \\ i \ge 2}} G(T_s,i,x_s,y) \Big\} + \\ + \min\Big\{ G(T_t,p-i,x,y),\ G(T_t,p-i,y,y),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t,p-i,x_t,y) \Big\}\Big\}. \quad (7)$$

2) $y \in T \setminus T_v$ and $y \neq x$.

$$G(T_v,p,x,y) = r(x,v) + \\ + \min_{0 \le i \le p} \Big\{ \min\Big\{ G(T_s,i,x,y),\ G(T_s,i,y,y),\ \min_{\substack{x_s \in T_s \\ i \ge 1}} G(T_s,i,x_s,y) \Big\} + \\ + \min\Big\{ G(T_t,p-i,x,y),\ G(T_t,p-i,y,y),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t,p-i,x_t,y) \Big\}\Big\}. \quad (8)$$

3) $y = x$.

$$\begin{aligned} G(T_v, p, x, x) &= r(x, v) + \\ &+ \min_{0 \le i \le p} \Big\{ \min \Big\{ G(T_s, i, x, x),\ \min_{\substack{x_s \in T_s \\ i \ge 1}} G(T_s, i, x_s, x) \Big\} + \\ &+ \min \Big\{ G(T_t, p-i, x, x),\ \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t, p-i, x_t, x) \Big\} \Big\}. \qquad (9) \end{aligned}$$

C) $x = y = v$ (Fig. 11-c). For this situation, we need a recurrence relation very similar to Case B3. The only difference is that we have to add the facility opening cost of v,

$$\begin{aligned} G(T_v, p, v, v) = r(v, v) + f(v) + \min_{0 \le i \le p-1} \Big\{ \min \Big\{ G(T_s, i, v, v),\ \min_{\substack{x_s \in T_s \\ i \ge 1}} G(T_s, i, x_s, v) \Big\} + \\ + \min \Big\{ G(T_t, p-1-i, v, v),\ \min_{\substack{x_t \in T_t \\ i \le p-2}} G(T_t, p-1-i, x_t, v) \Big\} \Big\}. \qquad (10) \end{aligned}$$

The base case for the recurrences defined above is easy to express. If v is a leaf in the tree and x and y are the facility serving v and the nearest facility respectively, the cost does not depend on y. We have

$$G(T_v, i, x, y) = r(x, v) + C, \qquad (11)$$

where $i = 0$ and $C = 0$ if $x \neq v$, and $i = 1$ and $C = f(v)$ if $x = y = v$.

4.3.2 Complexity analysis:

Based on the recurrence relations (3)-(10), we can prove the following complexity result.

Theorem 4.6. *The unrestricted k-median problem on trees is solved by the dynamic programming algorithm described above in $O(kn^3)$ time and $O(kn^2)$ space.*

Proof. We can directly apply Tamir's result for the classic k-median problem on trees [9] for which the number of discrete cost function values is $O(kn^2)$. In our case, the total number of discrete values that need to be computed for cost function G is $O(kn^3)$. Thus, we need to show that the recurrence relations can be implemented in constant amortized time. The space complexity of the algorithm is obviously $O(kn^2)$ because any time we compute the cost function for a vertex v we only need to keep the cost function for the children of v.

If we fix parameters x, y, and p in Equation (3), and if we focus only on a single distribution of facilities between T_s and T_t (a particular value for i), all the terms of (3) are available in constant time except for $A(i) = \min_{\substack{x_t \in T_t \\ i \le p-1}} G(T_t, p-i, x_t, y)$ which can be obtained in $O(|T_s|)$ time. However, $A(i)$ is used for the computation of the cost function $G(T_v, p, x, y)$ for all $x \in T$ and a fixed $y \in T$ and can be computed only once. Therefore, $A(i)$ is also available in amortized constant time for fixed x and y. The same argument can be made for relations (7) and (10). Hence, the analysis in [9] is applicable to our case. □

We conjecture that, for a fixed value of k, it is possible to use a continuous dynamic programming algorithm to solve the unrestricted k-median problem in time $O(n^2 \log^c n)$ where c is some constant, using an approach similar to that for the classic k-median problem of Benkoczi and Bhattacharya [2]. The key observation in favor of our claim is that $G(T_v, p, x, y)$ can be replaced by a function $G(T_v, p, x, \alpha)$ continuous in α if $y \in T \setminus T_v$. The parameter

α represents the distance from the nearest facility located outside of the subtree. In this way it is possible to reduce the complexity of all cost functions, continuous and discrete, to $O(n^2 \log^{c'} n)$ for some constant c' if the subtrees T_v are defined based on the spine decomposition [2]. This could possibly reduce the time complexity of the algorithm to $O(n^2 \log^c n)$.

5 Conclusion and Future Research

In this paper we study the collection depots location problem in trees both with the center and median objective functions. For the unrestricted 1-center problem we give an optimal $O(n)$ algorithm for both the discrete and continuous cases, improving the currently best known bound of $O(n \log n)$ of Tamir and Halman [10]. Their algorithm however is more general since it applies for restricted 1-center as well.

For the unrestricted 1-median problem, we propose an algorithm with time complexity $O(n \log n)$ and space complexity $O(n)$ which uses a recursive decomposition of trees called the spine decomposition [2]. We do not know if the restricted 1-median problem can be solved in sub-quadratic time. A trivial algorithm that runs in $O(n^2)$ time is to compute the cost of the restricted 1-median from all vertices in the tree.

Finally, we show that the restricted k-median problem is NP-complete even for star graphs if the cost of opening facilities is non-uniform. We also give an algorithm with $O(kn^3)$ time complexity and $O(kn^2)$ space complexity for the unrestricted k-median problem with non-uniform opening facility costs for tree graphs. If k is constant, we conjecture that the approach described in [2] for the classical k-median problem can be adapted for the CDFL problem as well, potentially reducing the time complexity to $O(n^2 \log^c n)$. We do not know if a further reduction in complexity by undiscretizing parameter x is possible or not.

References

[1] Robert Benkoczi. *Cardinality constrained facility location problems in trees.* PhD thesis, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 2004.

[2] Robert Benkoczi and Binay Bhattacharya. A new template for solving p-median problems for trees in sub-quadratic time. In G.S. Brodal and S. Leonardi, editors, *Proc. European. Symp. of Alg.*, volume LNCS 3669, pages 271–282, 2005.

[3] Robert Benkoczi, Binay Bhattacharya, Marek Chrobak, Lawrence Larmore, and Wojciech Rytter. Faster algorithms for k-median problems in trees. In B. Rovan and P. Vojtáš, editors, *Proc. 28th International Symposium on Mathematical Foundations of Computer Science*, volume LNCS 2747, pages 218–227, 2003.

[4] Oded Berman, Zvi Drezner, and George O. Wesolowsky. The collection depots location problem on networks. *Naval Research Logistics*, 49(1):15–24, 2002.

[5] Oded Berman and Rongbing Huang. Minisum collection depots location problem with multiple facilities on a network. *Journal of the Operational Research Society*, 55:769–779, 2004.

[6] Zvi Drezner and George O. Wesolowsky. On the collection depots location problem. *European Journal of Operational Research*, 130(3):510–518, 2001.

[7] Nimrod Megiddo. Linear time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12(4):759–776, 1983.

[8] Arnon Rosenthal and José A. Pino. A generalized algorithm for centrality problems on trees. *Journal of the ACM*, 36(2):349–361, 1989.

[9] Arie Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[10] Arie Tamir and Nir Halman. One-way and round-trip center location problems. *Discrete Optimization*, 2:168–184, 2005.