ELSEVIER

Operations Research Letters 30 (2002) 149–158

Operations Research Letters

www.elsevier.com/locate/dsw

Algorithmic results for ordered median problems

Jörg Kalcsics[a], Stefan Nickel[a], Justo Puerto[b], Arie Tamir[c,*]

[a] *Fraunhofer Institut für Techno- und Wirtschaftsmathematik, Kaiserslautern, Germany*
[b] *Facultad de Matemáticas, Universidad de Sevilla, Spain*
[c] *Department of Statistics & Operations Research, School of Mathematical Sciences, Tel Aviv, University, 69978 Ramat-Aviv, Israel*

Received 22 October 2001; received in revised form 8 March 2002; accepted 10 March 2002

Abstract

In a recent series of papers a new type of objective function in location theory, called ordered median function, has been introduced and analyzed. This objective function unifies and generalizes most common objective functions used in location theory. In this paper we identify finite dominating sets for these models and develop polynomial time algorithms together with a detailed complexity analysis. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Location theory; Finite dominating sets; Algorithms; Complexity

1. Introduction

In a recent series of papers a new type of objective function in location theory, called ordered median function, has been introduced and analyzed, see Francis et al. [8], Nickel [18], Nickel and Puerto [19], Puerto and Fernández [21,22] and Rodríguez-Chía et al. [23]. In this paper we develop algorithms for models using this objective function which is a generalization of the most popular objective functions: median, center, centdian, k-centrum, amongst many others (see Mirchandani and Francis [17] and Slater [24] for a description of these functions). We study the ordered median problem on several metric spaces: networks with positive and negative node weights, tree networks and the rectilinear space $\mathbb{R}^d$, $d \geqslant 2$. Moreover, we discuss the discrete versions and also the problem defined on directed networks.

Formally, the problem is defined as follows. Let X be a metric space equipped with a metric $d(.,.)$. Denote by $V = \{v_1, \ldots, v_n\}$, $V \subseteq X$, the set of existing facilities (demand points). Each v_i is associated with a weight w_i, $i = 1, \ldots, n$. Note that we do not require w_i to be positive. In those cases where w_i is negative we speak of obnoxious facilities [3]. Also given is a vector $\Lambda = (\lambda_1, \ldots, \lambda_n)$ with non-negative entries.

For each $x \in X$ define $d_i(x) := w_i d(x, v_i)$, $i = 1, \ldots, n$, $d(x) := (d_1(x), \ldots, d_n(x))$ and

$$d_{\leqslant}(x) := (d_{(1)}(x), \ldots, d_{(n)}(x)),$$

where $d_{(i)}(x)$ is the ith smallest element in the multi-set $\{w_i d(x, v_i)\}_{i=1}^n$.

The objective is to find $x \in X$ minimizing

$$M_\Lambda(x) := \sum_{i=1}^{n} \lambda_i d_{(i)}(x).$$

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

PII: S0167-6377(02)00121-9

Table 1
Summary of complexity results for ordered median problems

Complexity bounds					
	Networks				
	Undirected		Directed	Rectilinear $\mathbb{R}^d$	Discrete
	General	Trees			
General[a] ordered median	$O(mn^2 \log n)$	$O(n^3 \log n)$	$O(mn \log n)$	$O(n^{2d+1} \log n)$	$O(n^2 \log n)$
Concave ordered median	$O(n^2 \log n)$	$O(n^2)$[d]	$O(n^2 \log n)$		
Convex ordered median	$O(mn^2 \log n)$	$O(n \log^2 n)$	$O(n^2 \log n)$	$O(n \log^{2d} n)$	$O(n \log^2 n)$[f]
k-centrum[b]	$O(mn \log n)$[c]	$O(n \log n)$	$O(n^2 \log n)$	$O(n)$[e]	$O(n \log^2 n)$[g]

[a]Arbitrary node-weights.
[b]Non-negative node-weights.
[c]Unweighted. See [25].
[d]Unweighted.
[e]See [20].
[f]Discrete tree.
[g]Unweighted rectilinear planar case.

An optimal solution to this problem is called an *ordered median*. If $w_j \geqslant 0$, for $j = 1,\ldots,n$, and $\lambda_1 \leqslant \cdots \leqslant \lambda_n$ ($\lambda_1 \geqslant \cdots \geqslant \lambda_n$), we will call the model *convex* (*concave*). Notice that the classical center and median problems correspond, respectively, to the cases where $\Lambda = (0,\ldots,0,1)$, and $\Lambda = (1,\ldots,1,1)$. The centdian and the k-centrum models are derived by setting $\Lambda = (\mu,\ldots,\mu,1)$, and $\Lambda = (0,\ldots,0,1,\overset{k}{\ldots},1)$, respectively, ($\mu$ is a positive number bounded by 1.) Note that the four special cases are convex when $w_j \geqslant 0$, for $j = 1,\ldots,n$.

The rest of the paper is organized as follows. In Section 2 we study the single facility ordered median problem in general undirected and directed networks and present $O(mn^2 \log n)$ and $O(mn \log n)$ algorithms, respectively. For (undirected) trees the induced bound is $O(n^3 \log n)$. We show how to improve this bound to $O(n \log^2 n)$ in the convex case. Section 3 deals with the rectilinear ordered median problem in $\mathbb{R}^d$, for a fixed d. We present an $O(n \log^{2d} n)$ algorithm for the convex case. Finally, in Section 4, we develop some polynomial results on the discrete version of the ordered median model. Table 1 summarizes the results presented in this paper.

2. Finding the single facility ordered median of a general network

Let $G = (V,E)$ be an undirected graph with node set $V = \{v_1,\ldots,v_n\}$ and edge set $E = \{e_1,\ldots,e_m\}$. Each edge e_j, $j = 1,2,\ldots,m$, has a positive length l_j, and is assumed to be rectifiable. In particular, an edge $e_j := [v_r, v_s]$ is identified with an interval of length l_j so that we can refer to its interior points. Let $A(G)$ denote the continuum set of points on the edges of G. We view $A(G)$ as a connected and closed set which is the union of m intervals. Let $P[v_i, v_j]$ denote a simple path in $A(G)$ connecting v_i and v_j. We refer to interior points on an edge by their distances along the edge from the two nodes of the edge. The edge lengths induce a distance function on $A(G)$. For any pair of points $x, y \in A(G)$, we denote by $d(x, y)$ the length of a shortest path $P[x, y]$, connecting x and y. $A(G)$ is a metric space with respect to the above distance function. We refer to $A(G)$ as the network induced by G and the edge lengths $\{l_j\}$, $j = 1,\ldots,m$. A closed and connected subset of $A(G)$ which does not contain cycles is called a *subtree*. If a subtree is contained in an edge it is called a *sub-edge*.

In this section we deal with the case where $X = A(G)$. However, the definition below applies for any metric space X.

For all $v_i, v_j \in X$, $v_i \neq v_j$, $w_i w_j \neq 0$ define

$$EQ_{ij} := \{x \in X : w_i d(v_i, x) = w_j d(v_j, x)\}$$

and let EQ'_{ij} be the relative boundary of EQ_{ij}. Define $EQ := \bigcup_{\substack{i,j \\ i \neq j}} EQ'_{ij}$. The points in EQ are called *equilibrium points* of X. Note, for example, that in the planar rectilinear case the equilibrium set coincides with the boundary of the concept of a bisector. The properties of these sets are well-known. The interested reader is referred to [12,19,23].

A point x on an edge $e = [v_i, v_j] \in E$ is called a *bottleneck point* of node v_k, if $w_k \neq 0$, and

$$d(x, v_k) = d(x, v_i) + d(v_i, v_k) = d(x, v_j) + d(v_j, v_k).$$

Let BN_i denote the set of all bottleneck points of a node $v_i \in V$ and let $BN := \bigcup_{i=1}^{n} BN_i$ be the set of all bottleneck points of the graph.

Define $NBN := \bigcup_{\substack{i=1 \\ w_i < 0}}^{n} BN_i$. A point in NBN is called a *negative bottleneck point.*

To introduce our algorithmic results we first identify finite sets of points containing an ordered median. Such sets are called *finite dominating sets* (FDS) [11].

Nickel and Puerto [19] proved that $V \cup EQ$ is an FDS for the ordered median problem with non-negative weights. When some of the node weights are negative the results in [19] may not hold. However, it is possible to extend the result as follows:

Theorem 1. *The set $V \cup EQ \cup NBN$ is a finite dominating set for the single facility ordered median problem with general node weights.*

Proof. Let G be an undirected graph. Augment G by inserting the equilibria in EQ and negative bottleneck points from NBN as new nodes. $A(G)$ is now decomposed into sub-edges where each sub-edge connects two adjacent elements of $V \cup EQ \cup NBN$.

From the definition of equilibrium points, it follows that there exists a permutation of the weighted distance functions in $\{d_j(x)\}_{j=1}^{n}$ which is fixed for all the points x on every sub-edge. Therefore, the ordered median function reduces to the classical median function on every sub-edge. Since we include the negative bottleneck points NBN in the decomposition of the network, the distance functions are now piecewise linear and concave on every sub-edge. Therefore the desired result follows. □

We next show how to solve the ordered median problem on a general network. We solve the problem independently on each one of the edges, in $O(n^2 \log n)$ time. Restricting ourselves to a given edge $e_i = [v_s, v_t]$, from the above theorem we know that there exists a best point x^i with respect to the objective function, such that x^i is either a node, an equilibrium point or a negative bottleneck point. Hence, it is sufficient to calculate the objective at the two nodes of e_i, the set K_i of $O(n^2)$ equilibrium points, and the set L_i of $O(n)$ negative bottleneck points on e_i. (Note that if $w_j \geqslant 0$, $j = 1, \ldots, n$, we can ignore the bottleneck points.)

Generating the bottleneck and equilibrium points on e_i: Let x denote a point on e_i. (For convenience x will also denote the distance, along e_i, of the point from v_s.) For each $v_j \in V$, $d(x, v_j)$ is a piecewise linear concave function with at most one breakpoint. (If the maximum of $d(v_j, x)$ is attained at an interior point, this is a bottleneck point.) Assuming that all internodal distances have already been computed, it clearly takes constant time to construct $d_j(x)$ and the respective bottleneck point. If $w_j > 0$, $d_j(x)$ is concave and otherwise $d_j(x)$ is convex. To compute all the equilibrium points on e_i, we calculate in $O(n^2)$ total time the solutions to the equations $d_j(x) = d_k(x)$, where $v_j, v_k \in V$, $v_j \neq v_k$. To conclude, in $O(n^2)$ total time we identify the set L_i^* of $O(n)$ bottleneck points and the set K_i of $O(n^2)$ equilibrium points. Let $N_i = \{v_s, v_t\} \cup L_i^* \cup K_i$.

Computing the objective function at all points in N_i: First we sort in $O(|N_i| \log |N_i|)$ time the points in N_i. For any x in the interior of the sub-edge connecting x_q and x_{q+1}, where x_q, x_{q+1} are two consecutive elements in the sorted list of N_i, the order of $\{d_j(x)\}_{j=1}^{n}$ does not change. In particular, the objective value at x_{q+1} can be obtained from the objective value at x_q in constant time. The first point in the sorted list is $x = v_s$ and the objective value can be obtained in $O(n \log n)$ time. Therefore, the time needed to compute the objective for all points in N_i is $O(n \log n + |N_i| \log |N_i|)$. To summarize the total effort needed to compute a best

solution on e_i is

$$O(n^2 + n\log n + |N_i|\log|N_i|) = O(n^2 + |N_i|\log|N_i|).$$

The time to find a single facility ordered median of a network is therefore $O(mn^2 + mn^2\log n) = O(mn^2\log n)$.

The above complexity can be improved for some important special cases discussed in the literature. For example, if $w_j \geqslant 0$ for all $v_j \in V$, then we can disregard the bottleneck points although this does not improve the complexity. Moreover, if in addition $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$, then an optimal solution on e_i is attained either at v_s or v_t [19]. The objective at each node can be computed in $O(n\log n)$ and therefore, in this case, the optimal single facility ordered median point is obtained in $O(n^2\log n)$ time.

The ordered median problem on a directed graph can be treated similarly to the undirected case as the following analysis shows. Let $G_D = (V,E)$ be a (strongly connected) directed graph. Following Handler and Mirchandani [9], the weighted distance between a point $x \in A(G_D)$ and $v_i \in V$ is given by $\bar{d}(x,v_i) := w_i(d(x,v_i) + d(v_i,x))$. Denote $\bar{d}_i(x) := \bar{d}(x,v_i)$, for $i=1,\ldots,n$, and $\bar{d}(x) := (\bar{d}_1(x),\ldots,\bar{d}_n(x))$. The definitions of the other concepts carry over from the undirected case.

The ordered median problem on a directed network can be written as

$$\min_{x\in A(G_D)} M_\Lambda(x) := \sum_{i=1}^{n} \lambda_i \bar{d}_{(i)}(x). \tag{1}$$

First, we make some observations on the above distance functions.

Let $e=[v_i,v_j] \in E$ be an edge of the directed graph G_D, directed from v_i to v_j, and x a point in the interior of this edge. Then for a node $v_k \in V$, the distance function $\bar{d}_k(\cdot)$ is constant on the interior (v_i,v_j) of the edge $[v_i,v_j]$. Moreover, if $w_k \geqslant 0$ (respectively, $w_k < 0$) then $\bar{d}_k(v_i)$, $\bar{d}_k(v_j) \leqslant \bar{d}_k(x)$ (respectively, $\bar{d}_k(v_i)$, $\bar{d}_k(v_j) \geqslant \bar{d}_k(x)$) for all $x \in (v_i,v_j)$. With this observation we can derive a finite dominating set for this problem as in the undirected network case.

Theorem 2. *The ordered median problem on directed networks with non-negative node weights always has an optimal solution in the node-set V. If in addition $\lambda_1 > 0$ and $w_i > 0$, $\forall i = 1,\ldots,n$, then any optimal solution is in V.*

Proof. Let $e=[v_i,v_j] \in E$ and let x be an interior point of the edge $[v_i,v_j]$. It is sufficient to show that

$$\max\{M_\Lambda(v_i), M_\Lambda(v_j)\} \leqslant M_\Lambda(x) = \sum_{k=1}^{n} \lambda_k \bar{d}_{(k)}(x). \tag{2}$$

Without loss of generality, we prove $M_\Lambda(v_i) \leqslant M_\Lambda(x)$. Let us consider v_i and let $v_k \in V$ be an arbitrary node. From the observation above, we know that $\bar{d}_k(v_i) \leqslant \bar{d}_k(x)$, $k=1,\ldots,n$. Therefore, $\bar{d}(v_i) \leqslant \bar{d}(x)$, and by Theorem 1 in [8] also $\bar{d}_{\leqslant}(v_i) \leqslant \bar{d}_{\leqslant}(x)$. Eq. (2) follows by taking the scalar product with Λ.

Next, suppose that $\lambda_1 > 0$, and $w_j > 0$, for $j = 1,\ldots,n$. Therefore, we obtain $0 \leqslant d_{(j)}(v_i) \leqslant d_{(j)}(x)$, for $j = 1,\ldots,n$, and $0 = d_{(1)}(v_i) < d_{(1)}(x)$. Hence, since $\lambda_1 > 0$, we get $M_\Lambda(v_i) < M_\Lambda(x)$ and the result follows. □

From this result, the case of non-negative node weights can be solved in $O(n^2\log n)$ time by evaluating the function at each node of the network.

The case with positive and negative weights can also be solved by evaluating at most $O(m+n)=O(m)$ points in $A(G_D)$. (Since for strongly connected graphs $m \geqslant n$.) Indeed, since the functions $\bar{d}_k(x)$ are constant in the interior of an edge we only need to evaluate the objective function at an arbitrary interior point x_e of each edge $e \in E$. Based on the previous analysis we can solve the problem with positive and negative weights in $O(mn\log n)$ time.

We summarize the above results in the following Theorem.

Theorem 3. *An ordered median of an undirected (directed) network can be found in $O(mn^2\log n)$ ($O(mn\log n)$) time. Moreover, if $w_j \geqslant 0$, for all $v_j \in V$, and in addition for the undirected case $\lambda_1 \geqslant \cdots \geqslant \lambda_n \geqslant 0$, then the complexity bounds reduce to $O(n^2\log n)$.*

2.1. Finding the single facility ordered median of a tree

Throughout this section we will use the concept of convexity on trees as defined in Dearing et al. [6].

Rodríguez-Chía et al. [23] showed that when $w_i \geqslant 0$, for $i = 1, \ldots, n$, the ordered median function is convex on the plane with respect to any metric generated by norms, if the Λ-vector satisfies $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$. Next we prove an analogous result for trees.

Lemma 4. *Let $T = (V, E)$ be a tree network and $w_i \geqslant 0$, $i = 1, \ldots, n$. If $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$ then the function $M_\Lambda(\cdot)$ is convex on T.*

Proof. Let Λ satisfy $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$. By [6], $w_i d(x, v_i)$ is convex for $x \in A(T)$ and $i = 1, \ldots, n$. Denote by $\Pi(n)$ the set of all permutations of the set $\{1, \ldots, n\}$. Let $\sigma \in \Pi(n)$ be a fixed permutation of the set $\{1, 2, \ldots, n\}$ and $x \in A(T)$. The function $f_\sigma(x) = \sum_{i=1}^{n} \lambda_i w_{\sigma(i)} d(x, v_{\sigma(i)})$ is convex. Therefore, the function

$$g(x) = \max_{\tau \in \Pi(n)} \left\{ \sum_{i=1}^{n} \lambda_i w_{\tau(i)} d(x, v_{\tau(i)}) \right\}$$

is also convex as maximum of convex functions.

Since the λ_i's are non-decreasing, the permutation $\sigma \in \Pi(n)$, which sorts the weighted distance functions $w_i d(x, v_i)$ in the vector $d_{\leqslant}(x)$ for a given $x \in A(T)$, is identical to the permutation τ^*, which maximizes $g(x)$ for this x (see e.g. Theorem 368 in Hardy et al. [10]). Therefore, we obtain

$$M_\Lambda(x) = \max_{\tau \in \Pi(n)} \left\{ \sum_{i=1}^{n} \lambda_i w_{\tau(i)} d(x, v_{\tau(i)}) \right\}$$

and hence the desired result follows. □

For trees the same discretization results as for general networks hold with the additional simplification that we have no bottleneck points on trees and therefore $V \cup EQ$ is a finite dominating set for the problem with arbitrary weights.

From the analysis in the previous section (since $m = n - 1$), we can conclude that a best solution on a tree can be found in $O(n^3 \log n)$ time. Improvements are possible for some important cases.

2.1.1. The unweighted case: $w_j = w$ for all $v_j \in V$

In this case, each pair of distinct nodes, v_j, v_k contributes one equilibrium point. Moreover, such a point is the midpoint of the path $P[v_j, v_k]$, connecting v_j and v_k. Thus, $\sum_{e_i \in E} |N_i| = O(n^2)$. Let $T(n)$ denote the total time needed to find all the equilibrium points on the tree network. Then from the discussion on a general network we can conclude that the total time needed to solve the problem is $O(T(n) + n^2 \log n + \sum_{e_i \in E} |N_i| \log |N_i|) = O(T(n) + n^2 \log n)$.

We next show that $T(n) = O(n^2 \log n)$. More specifically, we show that with the centroid decomposition approach, in $O(\log n)$ time we can locate the equilibrium point defined by any pair of nodes $\{v_i, v_j\}$.

In the preprocessing phase we obtain in $O(n \log n)$ total time a centroid decomposition of the tree T [16]. In a typical step of this process we are given a subtree T' with q nodes, and we find, in $O(q)$ time an unweighted centroid, say v', of T', We also compute in $O(q)$ time the distances from v' to all nodes in T'. v' has the property that each one of the connected components obtained from the removal of v' from T' contains at most $q/2$ nodes.

Consider now a pair of nodes v_i and v_j and let r be a positive number satisfying $r \leqslant d(v_i, v_j)$. The goal is to find a point x on the path $P[v_i, v_j]$ whose distance from v_i is r. (The equilibrium point is defined by $r = d(v_i, v_j)/2$.) We use the above centroid decomposition recursively. First, we consider the centroid, say v_k, of the original tree T. Suppose that v_i is in a component T' and v_j is in a component T''. If $T' = T''$ we proceed recursively with T'. Otherwise, v_k is on $P[v_i, v_j]$. If $d(v_i, v_k) \geqslant r$, x is in T'. We proceed recursively with T', where the goal now is to find a point on the path $P[v_i, v_k]$ whose distance from v_i is r. If $d(v_i, v_k) < r$, x is in T'' and now recursively we proceed with T'' looking for a point on $P[v_k, v_j]$ whose distance from v_j is $d(v_i, v_j) - r$. This process terminates after $O(\log n)$ steps, each consuming constant effort. At the end we locate an edge containing the point x on $P[v_i, v_j]$ whose distance from v_i is exactly r. The point x is found in constant time solving the linear equation $d_i(x) = r$.

A further improvement is possible if we assume $w_j = w > 0$ for all $v_j \in V$ and $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n \geqslant 0$. From the previous section we know that in this case it is sufficient to compute the objective function at the nodes only, since there is an optimal solution which is a node. Then, we need to compute and sort, for each node v_j, the set of distances $\{d(v_j, v_k)\}_{v_k \in V}$. The total effort needed to obtain the n sorted lists of the distances

is $O(n^2)$ [13]. Therefore, in this case the problem is solvable in $O(n^2)$ time.

2.1.2. The convex case: $w_j \geqslant 0$ for all $v_j \in V$, and $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$

We first solve this case on a path graph in $O(n \log^2 n)$ time. For a path graph we assume that the nodes in V are points on the real line. Using the convexity of the objective, we first apply a binary search on V to identify an edge (a pair of adjacent nodes), $[v_i, v_{i+1}]$, containing an ordered median x. Since it takes $O(n \log n)$ time to evaluate the objective at any point v_j, the edge $[v_i, v_{i+1}]$ is identified in $O(n \log^2 n)$ time. Restricting ourselves to this edge we note that for $j = 1, \ldots, n$, $w_j|x - v_j|$ is a linear function of the parameter x.

To find an optimum, x^*, we use the general parametric procedure of Megiddo [15], with the modification in Cole [5]. The reader is referred to these references for a detailed discussion of the parametric approach. We only note that the master program that we apply is the sorting of the n linear functions, $\{w_j|x - v_j|\}$, $j = 1, \ldots, n$ (using x as the parameter). The test for determining the location of a given point x' w.r.t. x^* is based on calculating the objective $M_\Lambda(x)$ and determining its one-sided derivatives at x'. This can clearly be done in $O(n \log n)$ time. We now conclude that with the above test, the parametric approach in [15,5] will find x^* in $O(n \log^2 n)$ time.

We now turn to the case of a general tree. As shown above, in this case the objective function is convex on any path of the tree network. We will use a binary search (based on centroid decomposition) to identify an edge containing an optimal solution in $O(\log n)$ phases.

In the first phase of the algorithm we find, in $O(n)$ time, an unweighted centroid of the tree, say v_k. Each one of the connected components obtained by the removal of v_k contains at most $n/2$ nodes. If v_k is not an optimal ordered median, then due to the convexity of the objective, there is exactly one component, say T^j, such that any optimal solution is either in that component or on the edge connecting the component to v_k. We proceed to search in the subtree induced by v_k and T^j. Since T^j contains at most $n/2$ nodes, this search process will have $O(\log n)$ phases, when at the end an edge containing an optimal solution is identified. To locate the ordered median on an edge we use the above $O(n \log^2 n)$ for path trees.

To evaluate the total complexity of the above algorithm we now analyze the effort spent in each one of the $O(\log n)$ phases. At each such iteration a node (centroid) v_k is given and the goal is to check the optimality of v_k and identify the (unique) direction of improvement if v_k is not optimal. To facilitate the discussion, let $\{v_{k(1)}, \ldots v_{k(l)}\}$ be the set of neighbors of v_k. To test optimality it is sufficient to check the signs of the derivatives of the objective at v_k in each one of the l directions. (There is at most one negative derivative.)

We first compute and sort in $O(n \log n)$ time the multi-set $\{w_j d(v_j, v_k)\}$ of weighted distances of all nodes of the tree from v_k. Let L^k denote this sorted list. (We also assume that the weights $\{w_i\}_{i=1,\ldots,n}$ have already been sorted in a list W.)

Suppose first that all the elements in L^k are distinct. We refer to this case as the non-degenerate case. Let σ denote the permutation of the nodes corresponding to the ordering of L^k, i.e.,

$$w_{\sigma(1)} d(v_{\sigma(1)}, v_k) < \cdots < w_{\sigma(n)} d(v_{\sigma(n)}, v_k).$$

Then the derivative of the objective at v_k in the direction of its neighbor $v_{k(l)}$ is given by

$$-\sum_{v_{\sigma(i)} \in T^{k(l)}} \lambda_i w_{\sigma(i)} + \sum_{v_{\sigma(i)} \in V \setminus T^{k(l)}} \lambda_i w_{\sigma(i)}.$$

Equivalently, the derivative is equal to

$$\sum_{i=1}^{n} \lambda_i w_{\sigma(i)} - 2 \sum_{v_{\sigma(i)} \in T^{k(l)}} \lambda_i w_{\sigma(i)}.$$

It is therefore clear that after the $O(n \log n)$ effort needed to find σ, we can compute all l directional derivatives in $O(n)$ time.

Next we consider the case where the elements in L^k are not distinct. Assume without loss of generality that $w_j > 0$, for $j = 1, \ldots, n$. In this case we partition the node set into equality classes, $\{U^1, \ldots, U^p\}$, such that for each $q = 1, \ldots, p$, $w_j d(v_j, v_k) = c^q$ for all $v_j \in U^q$, and $c^1 < c^2 < \cdots < c^p$. (Note that $c^1 = 0$ and $v_k \in U^1$.)

Consider an arbitrary perturbation where we add ε^j to the length of edge e_j, $j = 1, \ldots, m$. Let d' denote the distance function on the perturbed tree

network. If ε is sufficiently small all the elements in the set $\{w_j d'(v_j, v_k)\}$ are distinct, and for each pair of nodes, v_s, v_t with $v_s \in U^q$ and $v_t \in U^{q+1}$, $q = 1, \ldots, p-1$, $w_s d'(v_s, v_k) < w_t d'(v_t, v_k)$. Therefore, as discussed above, in $O(n \log n)$ time we can test whether v_k is optimal with respect to the perturbed problem, and if not find the (only) neighbor of v_k, say $v_{k(t)}$, such that the derivative in the direction of $v_{k(t)}$ is negative.

In the next lemma, we will prove that if v_k is optimal for the perturbed problem it is also optimal for the original problem. Moreover, if v_k is not optimal for the original problem and $v_i = v_{k(t)}$ is a neighbor defining a (unique) direction of improvement for the original problem, then it also defines the (unique) improving direction for the perturbed problem.

This result will imply that in $O(n \log n)$ time we can test the optimality of v_k for the original problem, and identify an improving direction if v_k is not optimal.

Lemma 5. *Let $v_i = v_{k(t)}$ be a neighbor of v_k, and let A_i and $A_i(\varepsilon)$ denote the derivatives of the objective $M_A(x)$ at v_k in the direction of v_i for the original and the perturbed problems, respectively. Then $A_i(\varepsilon) \leqslant A_i$.*

Proof. Because of the additivity property of directional derivatives it is sufficient to prove the result for the case where V is partitioned into exactly two equality classes, U^1, U^2, where $U^1 = \{v_k\}$, $c^1 = 0$, and $w_j d(v_j, v_k) = c^2$ for all $v_j \neq v_k$.

Consider an arbitrary neighbor $v_i = v_{k(t)}$ of v_k, and let T^i be the component of T, obtained by removing v_k, which contains v_i. Let n_i denote the number of nodes in T^i. Let τ denote the permutation arranging the n_i nodes in T^i in a non-increasing order of their weights, and the remaining $n - n_i$ nodes in a non-decreasing order of their weights. We have $v_{\tau(t)} \in T^i$, $t = 1, \ldots, n_i$, $v_{\tau(t)} \notin T^i$, $t = n_i + 1, \ldots, n$, $w_{\tau(1)} \geqslant w_{\tau(2)} \geqslant \cdots \geqslant w_{\tau(n_i)}$, and $w_{\tau(n_i+1)} \leqslant w_{\tau(n_i+2)} \leqslant \cdots \leqslant w_{\tau(n)}$. Then, it is easy to verify that

$$A_i = -\sum_{t=1}^{n_i} \lambda_t w_{\tau(t)} + \sum_{t=n_i+1}^{n} \lambda_t w_{\tau(t)}.$$

Next, to compute $A_i(\varepsilon)$, consider the perturbed problem and the permutation σ arranging the nodes in an increasing order of their distances $\{w_j d'(v_j, v_k)\}$. Specifically, $0 = w_k d'(v_k, v_k) = w_{\sigma(1)} d'(v_{\sigma(1)}, v_k)$, and

$$w_{\sigma(1)} d'(v_{\sigma(1)}, v_k) < \cdots < w_{\sigma(n)} d'(v_{\sigma(n)}, v_k).$$

With the above notation it is easy to see that

$$A_i(\varepsilon) = -\sum_{j | v_{\sigma(j)} \in T^i} \lambda_j w_{\sigma(j)} + \sum_{j | v_{\sigma(j)} \notin T^i} \lambda_j w_{\sigma(j)}.$$

We are now ready to prove that $A_i(\varepsilon) \leqslant A_i$.

We will successively bound $A_i(\varepsilon)$ from above as follows: suppose that there is an index j such that $v_{\sigma(j)} \in T^i$, and $j > n_i$, i.e., there exists a node $v_t \in T^i$ such that in the expression defining $A_i(\varepsilon)$, w_t is multiplied by $-\lambda_j$ for some $j > n_i$. Since $|T^i| = n_i$, there exists a node $v_s \notin T^i$ such that w_s is multiplied by λ_b for some index $b \leqslant n_i$, in this expression. Thus, from $b \leqslant n_i < j$, we have $\lambda_b \leqslant \lambda_j$, which in turn yields

$$-\lambda_j w_t + \lambda_b w_s \leqslant -\lambda_b w_t + \lambda_j w_s.$$

The last inequality implies that if we swap w_s and w_t, multiplying the first by λ_j and the second by $-\lambda_b$ we obtain an upper bound on $A_i(\varepsilon)$. Applying this argument successively and swapping pairs of nodes, as long as possible, we conclude that there is a permutation σ' of the nodes in V, such that each node $v_{\sigma'(j)} \in T^i$ is matched with $-\lambda_j$, for some $1 \leqslant j \leqslant n_i$; each node $v_{\sigma'(j)} \notin T^i$ is matched with λ_j, for some $n_i < j \leqslant n$, and

$$A_i(\varepsilon) \leqslant -\sum_{j=1}^{n_i} \lambda_j w_{\sigma'(j)} + \sum_{j=n_i+1}^{n} \lambda_j w_{\sigma'(j)}.$$

Denote the right-hand side of the last inequality by $B_i^{\sigma'}(\varepsilon)$.

To conclude the proof consider the entire collection of all permutations σ'', which assign every node in T^i to a (unique) index $1 \leqslant j \leqslant n_i$, and every node which is not in T^i to a (unique) index $n_i < j \leqslant n$. For each such permutation consider the respective expression

$$B_i^{\sigma''}(\varepsilon) = -\sum_{j=1}^{n_i} \lambda_j w_{\sigma''(j)} + \sum_{j=n_i+1}^{n} \lambda_j w_{\sigma''(j)}.$$

From Theorem 368 in Hardy et al. [10], mentioned above, the maximum of the above expression over all such permutations is achieved for the permutation τ, which arranges the n_i nodes in T^i in a non-increasing order of their weights, and the remaining $n - n_i$ nodes

not in T^i in a non-decreasing order of their weights. We note that this maximizing permutation is exactly the one defining A_i. Therefore,

$$A_i(\varepsilon) \leqslant B_i^{\sigma'}(\varepsilon) \leqslant B_i^{\tau}(\varepsilon) = A_i. \qquad \square$$

To summarize, to check the optimality of a node v_k in the original problem, we can use the following procedure:

Compute and sort elements in set $\{w_j d(v_j, v_k)\}$. From the sorted sequence define the equality classes $\{U^1, \ldots, U^p\}$. Arbitrarily select an ordering (permutation) of the nodes in V, such that for any $q = 1, \ldots, p-1$, and any pair of nodes $v_j \in U^q$, $v_t \in U^{q+1}$, v_j precedes v_t. Let σ denote such a selected permutation. The permutation defines a perturbed problem, which in turn corresponds to a non-degenerate case. Therefore, as noted above, in $O(n \log n)$ time we can check whether v_k is optimal for the perturbed problem. If v_k is optimal for the perturbed problem, i.e., $A_i(\varepsilon) \geqslant 0$ for each neighbor v_i, by the above lemma it is also optimal for the original problem. Otherwise, there is a unique neighbor of v_k, say $v_i = v_{k(t)}$ such that the derivative of the perturbed problem at v_k in the direction of v_i is negative. From Lemma 5 to test optimality of v_k for the original problem, it is sufficient to check only the sign of the derivative of the original problem at v_k in the direction of v_i. The latter step can be done in $O(n)$ time.

To summarize, the algorithm has $O(\log n)$ phases, where in each phase we spend $O(n \log n)$ time to test the optimality of some node, and identify the unique direction of improvement if the node is not optimal. At the end of this process an edge of the given tree which contains an optimal solution is identified. As explained above, the time needed to find an optimal solution on an edge is $O(n \log^2 n)$ time. Therefore, the total time to solve the problem is $O(n \log^2 n)$.

We note in passing that the above approach is applicable to the special case of the k-centrum problem. Since testing optimality of a node for this model will take only $O(n)$ time, the total time for solving the k-centrum problem will be $O(n \log n)$. This bound improves upon the complexity given in Tamir [25] by a factor of $O(\log n)$.

The next theorem summarizes the complexity results for tree graphs we have obtained above.

Theorem 6. *An ordered median of an undirected tree can be computed in $O(n^3 \log n)$ time.*

1. *In the unweighted case, i.e., $w_j = w$ for all $v_j \in V$, the time is $O(n^2 \log n)$. If, in addition, $w > 0$, and $\lambda_1 \geqslant \cdots \geqslant \lambda_n \geqslant 0$, the time is further reduced to $O(n^2)$.*
2. *If $w_j \geqslant 0$, for all $v_j \in V$, and $0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_n$, the ordered median can be found in $O(n \log^2 n)$ time.*

3. Finding the ordered median in the rectilinear space

In this case the metric space is $X = \mathbb{R}^d$ equipped with the rectilinear metric, where $d(x, y) = \|x - y\|_1 := \sum_{i=1}^{d} |x_i - y_i|$.

Suppose that d is fixed, and consider first the general case of the ordered median. The collection consisting of the $O(n^2)$ (piecewise linear) bisectors $\{w_j d(x, v_j) - w_i d(x, v_i) = 0\}$, $i, j = 1, \ldots, n$, and the $O(n)$ hyperplanes, which are parallel to the axes and pass through $\{v_1, \ldots, v_n\}$ induces a cell partition of $\mathbb{R}^d$. This partition can be computed in $O(n^{2d})$ time for any fixed $d \geqslant 2$ (see Edelsbrunner [7]). If there is a finite ordered median, then at least one of the $O(n^{2d})$ vertices of the partition is an ordered median. (Notice that only if there are some negative weights finite ordered medians may not exist.) Hence, by evaluating the objective at each vertex and each infinite ray of the partition we solve the problem in $O(n^{2d+1} \log n)$ time.

In the convex case in $\mathbb{R}^d$ we can directly use the approach of Cohen and Megiddo [4] to get a complexity of $O(n \log^{2d+1} n)$. This approach relies only on the fact that the objective function can be evaluated at any point just using additions, multiplications by a scalar and comparisons. Clearly, the ordered median objective function in this case falls into this class. More precisely, the complexity analysis involves several components which we now discuss. First, we have to give a bound T on the number of operations needed to evaluate the objective function at a given point. In the case of ordered median functions this is $O(n \log n)$. The number of comparisons to be performed is $O(n \log n)$ and this can be done in $r = O(\log n)$ parallel phases using $C_i = O(n)$ effort in each phase [1]. Then the result

by [4] states that the bound to find an optimal solution is $O(d^3 T(\sum_{i=1}^{r} \lceil \log C_i \rceil)^d)$ for a fixed dimension d. In our case we achieve the bound $O(n \log^{2d+1} n)$. (The same bound can also be achieved by using the results in Tokuyama [26].) The bound in [4] is achieved by (recursively) solving $O(\log^2 n)$ recursive calls to instances of lower dimension. For $d = 1$ the above general bound (applied to our problem) gives $O(n \log^3 n)$. However, note that for our problem we actually solve the case $d = 1$ in Section 2.1.2 in $O(n \log^2 n)$ time, an improvement by a factor of $\log n$. Therefore, for any fixed $d \geqslant 2$, the bound will be reduced by a factor of $\log n$, and we have,

Theorem 7. *Suppose that $w_j \geqslant 0$, for all $v_j \in V$, and $0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_n$. Then for any fixed d, the (convex) rectilinear ordered median problem can be solved in $O(n \log^{2d} n)$ time.*

For comparison purposes, we note that the important special case of the k-centrum functions ($\Lambda = (0, \ldots, 0, 1, \ldots, 1)$) has been recently solved in $O(n)$ time for any fixed d, in Ogryczak and Tamir [20].

We briefly mention that the results on the rectilinear model presented above can be extended to the more general case where the rectilinear norm is replaced by any polyhedral norm where the number of extreme points of the unit ball is constant.

4. The discrete model

We have considered above the case where the ordered median can be located anywhere in the metric space. Such a location model is referred to as a continuous model. In a discrete version of the problem, the ordered median is restricted to some discrete set, commonly the set V consisting of the existing facilities. Clearly, these discrete problems can be solved by evaluating the objective function at each one of the points in the discrete set. If the discrete set is V and the distances between the points in V are given, the total effort to solve the problem is $O(n^2 \log n)$. For some special cases this bound can be improved. For example, the solution to the discrete convex tree case discussed above is attained at one of the nodes of an edge containing the continuous solution. Therefore, the continuous and the discrete solutions are derived using the same computational effort, namely $O(n \log^2 n)$.

It is not yet known whether we can get sub-quadratic time even for the discrete version of the continuous convex rectilinear planar problem discussed above. For the special case of the discrete unweighted k-centrum problem ($w_j = w$, $j = 1, \ldots, n$), an $O(n \log^2 n)$ algorithm can be obtained as follows.

If $w < 0$ we use the results of Bespamyatnikh et al. [2] to compute in $O(n \log^2 n)$ the sum of the closest k points in V to each one of the n points. The optimal value for the discrete problem is attained at that point where the sum is largest. If $w > 0$, first compute in $O(n \log n)$ total time, for each point the sum of its distances to all other points [14]. Next using [2], compute in $O(n \log^2 n)$ for $p = n - k$, the sum of the closest p points in V to each one of the n points. Then, by subtracting we have for each point in V, the sum of its distances to the furthest k points. The optimal value for the discrete problem is attained at that point where the sum is smallest.

References

[1] M. Ajtai, J. Komlos, E. Szemeredi, Sorting in $c \log n$ parallel steps, Combinatorica 3 (1983) 1–19.
[2] S. Bespamyatnikh, K. Kedem, M. Segal, A. Tamir, Optimal facility location under various distance functions, Int. J. Comput. Geom. Appl. 10 (2000) 523–534.
[3] R.E. Burkard, J. Krarup, A linear algorithm for the pos/neg-weighted 1-median problem on a cactus, Computing 60 (1998) 193–215.
[4] E. Cohen, N. Megiddo, Maximizing concave functions in fixed dimension, in: P.M. Pardalos (Ed.), Complexity in Numerical Optimization, World Scientific, Singapore, 1993, pp. 74–87.
[5] R. Cole, Slowing down sorting networks to obtain faster algorithms, J. Assoc. Comput. Math. 34 (1987) 168–177.
[6] P.M. Dearing, R.L. Francis, T.J. Lowe, Convex location problems on tree networks, Oper. Res. 24 (1976) 628–642.
[7] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer, Berlin, 1987.
[8] R.L. Francis, T.J. Lowe, A. Tamir, Aggregation error bounds for a class of location models, Oper. Res. 48 (2000) 294–307.
[9] G.W. Handler, P.B. Mirchandani, Location on Networks: Theory and Algorithms, MIT Press, Cambridge, MA, 1979.
[10] G.H. Hardy, J.E. Littlewood, G. Polya, Inequalities, Cambridge University Press, Cambridge, 1952.
[11] J.N. Hooker, R.S. Garfinkel, C.K. Chen, Finite dominating sets for network location problems, Oper. Res. 39 (1991) 100–118.

[12] C. Icking, R. Klein, L. Ma, S. Nickel, A. Weissler, On bisectors for different distance functions, Discrete Appl. Math. 109 (2001) 139–161.
[13] T.U. Kim, T.J. Lowe, A. Tamir, J.E. Ward, On the location of a tree-shaped facility, Networks 28 (1996) 167–175.
[14] Y. Konforty, A. Tamir, The rectilinear single facility location problem with minimum distance constraints, Location Sci. 5 (1998) 147–163.
[15] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, J. Assoc. Comput. Math. 30 (1983) 852–865.
[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] P.B. Mirchandani, R.L. Francis (Eds.), Discrete Location Theory, Wiley, New York, 1990.
[18] S. Nickel, Discrete ordered Weber problems, in: Operations Research Proceedings 2000, B. Fleischmann, R. Lasch, U. Derigs, W. Domschke and U. Rieder (Eds.), Springer, Berlin, 2001, pp. 71–76.
[19] S. Nickel, J. Puerto, A unified approach to network location problems, Networks 34 (1999) 283–290.
[20] W. Ogryczak, A. Tamir, Minimizing the sum of the k largest functions in linear time, Technical Report, Tel Aviv University, Tel Aviv, Israel, 2001.
[21] J. Puerto, F.R. Fernández, The symmetrical single facility location problem. Technical Report, Prepublicación de la Facultad de Matemáticas, Universidad de Sevilla, 1995.
[22] J. Puerto, F.R. Fernández, Geometrical properties of the symmetrical single facility location problem, J. Nonlinear Convex Anal. 1 (2000) 321–342.
[23] A.M. Rodríguez-Chía, S. Nickel, J. Puerto, F.R. Fernández, A flexible approach to location problems, Math. Methods Oper. Res. 51 (2000) 69–89.
[24] P. Slater, Center to centroids in graphs, J. Graph Theory 2 (1978) 209–222.
[25] A. Tamir, The k-centrum multi-facility location problem, Discrete Appl. Math. 109 (2001) 293–307.
[26] T. Tokuyama, Minimax parametric optimization problems in multi-dimensional parametric searching, in: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 75–84.