Available online at www.sciencedirect.com

SCIENCE DIRECT®

Operations Research Letters 32 (2004) 249–257

Operations Research Letters

www.elsevier.com/locate/dsw

Sorting weighted distances with applications to objective function evaluations in single facility location problems

Arie Tamir

School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel

Received 27 April 2003; received in revised form 18 September 2003; accepted 2 October 2003

Abstract

We consider single facility location problems defined on rectilinear spaces and spaces induced by tree networks. We focus on discrete cases, where the facility is restricted to be in a prespecified finite set S, and the goal is to evaluate the objective at each point in S. We present efficient improved algorithms to perform this task for several classes of objective functions.

Keywords: Facility location; Weighted distances; Ordered-median objective

1. Introduction

In a typical single facility location problem we are given a finite set of points $V = \{v_1, \ldots, v_n\}$ in some metric space X, and the goal is to find a point $x \in X$ which minimizes some objective function of the distances of the points in V from x. V is viewed as the set of customers or demand points, and x is the location of the serving facility, or server. For each pair of points $u, v \in X$ we let $d(u, v)$ denote the (symmetric) distance between u and v. Classical examples are the weighted 1-center and 1-median problems, defined by the following objective functions. Suppose that each point v_j, $j = 1, \ldots, n$, is associated with a nonnegative weight w_j. (By an *unweighted model* we will refer to the case, where $w_j = 1$, $j = 1, \ldots, n$.) Weighted models have been quite standard in location theory, (see, for example [16,17]). If $d(v_j, x)$ is the (service) distance of customer v_j to the server located at x, then $w_j d(v_j, x)$ can be interpreted as the travel time of v_j to this server. In the 1-center model the goal is to minimize the objective function $\max\{w_j d(v_j, x) : j = 1, \ldots, n\}$, while in the 1-median problem we wish to minimize the function $\sum_{j=1}^{n} w_j d(v_j, x)$.

A more general and unifying model is the *ordered-median* function, (see [14] and the references cited therein), defined as follows.

Given is a real vector $\Lambda = (\lambda_1, \ldots, \lambda_n)$. For any $x \in X$ consider the multi-set of weighted distances from x, $D(x) = \{w_1 d(v_1, x), \ldots, w_n d(v_n, x)\}$. Define the real vector $D^*(x) = (d^1(x), \ldots, d^n(x))$, where $d^j(x)$, $j = 1, \ldots, n$, is a jth largest element in the set $D(x)$. The value of the (Λ) ordered-median function at x is given by

$$\sum_{j=1}^{n} \lambda_j d^j(x).$$

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

Note that the center and the median objectives mentioned above, are defined by $\Lambda = (1, 0, \dots, 0)$ and $\Lambda = (1, 1, \dots, 1)$, respectively. There are two other important special cases discussed in the literature. The first is the *k-centrum* objective [24,30], $H_k(x)$, defined by the vector Λ which has the first k components equal to 1, and the last $n-k$ components equal to 0, i.e.,

$$H_k(x) = \sum_{j=1}^{k} d^j(x).$$

The second is the *lexicographic center* objective, which can be characterized by an instance of Λ, where λ_j is 'significantly' larger than λ_{j+1}, for $j = 1, \dots, n-1$, and $\lambda_j \geqslant 0$, for $j = 1, \dots, n$, [10].

Generally, suppose that we are given a real function g, defined on R^n, and the objective function of the single facility location problem is $f(x) = g(D^*(x))$. When there are no restrictions on x, the location of the server, we refer to the location model as a *continuous* version.

We focus here on *discrete* models, where the server must be located at some point x of a prespecified finite set $S = \{u_1, \dots, u_m\} \subset X$, which we call the *feasible set*. In the discrete case we can use a graph to represent the model. Let G be a complete undirected graph with the node set $V \cup S$. For each pair of nodes (points) $p, q \in V \cup S$, the weight of the edge (p, q) is $d(p, q)$, the symmetric distance between the pair in the metric space X. Also, as stated above, each node $v_j \in V$ is associated with a nonnegative weight w_j. We then view $f(x) = g(D^*(x))$ as a real function defined on S. If the goal is only to find the minimum value of $f(x)$ on S, then in some special cases, e.g., the rectilinear planar 1-center and 1-median problems, we can exploit the special structure and the convexity of the objective to find the optimal point without explicitly evaluating the objective at each point of S. But in the general case, we will need to evaluate $f(x)$ at each point $x \in S$ in order to identify the minimum value. We refer to this task of computing the objective function at all points in S as a *complete evaluation*.

Another context where complete evaluation is often used is multi-objective optimization [4,11,28]. To illustrate and motivate consider the bi-objective case, with two real objective functions, $f_1(x) = g_1(D^*(x))$ and $f_2(x) = g_2(D^*(x))$, (e.g., the 1-center and the 1-median objectives), defined on S. A key concept in bi-objective minimization is the *pareto set*, which consists of all non-dominated solutions in S. Specifically, consider the planar set $S(f_1, f_2) = \{(f_1(x), f_2(x)) : x \in S\}$. A point $(a, b) \in S(f_1, f_2)$ is dominated if there is a point $(c, d) \in S(f_1, f_2)$, such that $c \leqslant a$, $d \leqslant b$ and $(a, b) \neq (c, d)$. A common approach to generate the pareto set is to apply complete evaluation to both f_1 and f_2 and then eliminate the dominated solutions. (Note that the elimination phase can be executed in $O(m \log m)$ time [15,25].)

In this communication, we consider the complete evaluation of the single facility objective function $f(x) = g(D^*(x))$, defined above. We make the following suppositions. First, suppose that for each pair of points u, v of the metric space X, it takes constant time to compute their distance $d(u, v)$. Second, given a real vector $(z_1, \dots, z_n)$, it takes $O(n)$ time to compute $g(z_1, \dots, z_n)$. Therefore, by a straightforward approach, for any $x \in S$ the multi-set $D(x)$ is computable in $O(n)$ time, and the respective real vector $D^*(x)$ can be generated in $O(n \log n)$ time. As a result, assuming that $S = \{u_1, \dots, u_m\}$, a complete evaluation of $f(x)$ over S will take $O(mn \log n)$ time. The question is whether we can improve upon the latter complexity bound which is determined by the effort to generate all the vectors of sorted weighted distances, $D^*(x)$, $x \in S$. Note that a trivial lower bound on the effort to perform this task is $\Omega(mn)$, since $|S| = m$, and for each $x \in S$, $D^*(x)$ has n components. We will prove that for rectilinear spaces and spaces induced by tree networks, all these vectors can be computed in $O(n^2 + mn)$ and $O((m+n)^2)$ times, respectively. In the classical discrete location models considered in the literature, it is commonly assumed that a server can be located only at points in V, i.e., $S = V$, see for example [16,17]. Hence, for this case we reduce the complexity by a factor of $O(\log n)$. (We will concentrate mainly, but not exclusively, on this case.)

We are unaware of papers in the literature that suggest efficient implementation of complete evaluation in the context of functions defined in terms of weighted distances. To the best of our knowledge there are results for the unweighted case only. For example, in the unweighted case, [29] presents an $O(n^2)$ algorithm to compute all the vectors $D^*(x)$, $x \in S$, for a tree graph when $S = V$. The weighted case is considerably more complex. To illustrate, consider the case where $V = \{v_1, \dots, v_n\}$ is a set of points on the real line and $S = V$. In the unweighted case, we first sort the points in V, and assume without loss of generality that $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_n$. Then, for each point v_i, the vector $D^*(v_i)$ can be generated in $O(n)$ time by merging the two (sorted) lists $(v_i - v_1, \ldots, v_i - v_i)$ and $(v_{i+1} - v_i, \ldots, v_n - v_i)$ [1]. It is not obvious what preprocessing and tools should be used to extend the above to the weighted case in order to achieve the $O(n^2)$ total complexity bound. We discuss this case and its extension to general rectilinear spaces in Section 2. In Section 3, we consider tree networks.

In addition to [29] there are also some isolated results in the literature which refer to specific objective functions. We will cite and discuss them later, when we study the relevant weighted models in Sections 3 and 4.

In Section 4, we consider some important special cases where a complete evaluation of the objective function can be performed in subquadratic time. Some relevant open problems are listed in the last section.

2. Sorting in rectilinear spaces

Consider first the case, where $V = \{v_1, \ldots, v_n\}$ and $S = \{u_1, \ldots, u_m\}$ are sets of points on the real line and $d(u_i, v_j) = |u_i - v_j|$. For each $i = 1, \ldots, m$, the multi-set $D(u_i)$ is given by

$$D(u_i) = \{w_j|v_j - u_i| : j = 1, \ldots, n\}.$$

For $i = 1, \ldots, m$, we now define the super multi-set

$$\hat{D}(u_i) = \{w_j(v_j - u_i) : j = 1, \ldots, n\}$$
$$\cup \{-w_j(v_j - u_i) : j = 1, \ldots, n\}.$$

We first note that for each $i = 1, \ldots, m$, $\hat{D}(u_i)$ contains at most n positive entries and at most n negative entries. The multi-set $D(u_i)$ consists of the largest n elements of $\hat{D}(u_i)$. Therefore, in order to justify the claim that all the vectors $D^*(u_i)$, $i = 1, \ldots, m$, can be generated in $O(n^2 + mn)$ time, it is sufficient to show that the total effort to sort (individually) all the super multi-sets $\hat{D}(u_i)$, $i = 1, \ldots, m$, is $O(n^2 + mn)$.

We embed this problem in a planar setup. Consider the collection L of the $2n$ lines in the plane defined by the equations $\{y = w_j(v_j - x) : j = 1, \ldots, n\}$ and $\{y = -w_j(v_j - x) : j = 1, \ldots, n\}$.

The elements of $\hat{D}(u_i)$ correspond to the intersection points of the vertical line $x = u_i$ with the lines in L. Therefore, we can use the machinary developed in [6,7], to sort the above multi-sets.

Let L' be the collection of the m lines in the plane defined by the equations $\{x = u_i : i = 1, \ldots, m\}$.

We first construct the planar arrangement of the collection L in $O(n^2)$ time, using the incremental algorithm in [6,7]. There are $2n$ stages, where at each stage we augment a new line from L, and obtain in $O(n)$ time the sorted sequence of intersection points of the new line with the lines that have already been introduced.

We then consider the collection L'. For each vertical line $x = u_i$ in L' we apply (individually) an additional single stage of the incremental algorithm, and obtain in $O(n)$ time L_i, the sorted sequence of the y values of the intersection points of this line with the lines in L. The vector $D^*(u_i)$ consists of the largest n elements in L_i.

To conclude, we have shown that for the case of the real line the total effort to generate all the vectors $D^*(u_i)$, $u_i \in S$, is $O(n^2 + mn)$.

We now show how to extend the case of the real line to the general rectilinear case. To facilitate the discussion, suppose that for $j = 1, \ldots, n$, $v_j = (v_j^1, \ldots, v_j^d) \in R^d$, and for $i = 1, \ldots, m$, $u_i = (u_i^1, \ldots, u_i^d) \in R^d$. Define

$$\Delta = \{\delta = (\delta_1, \ldots, \delta_d) : \delta_k \in \{+1, -1\},$$
$$k = 1, \ldots, d\}.$$

For each pair u_i, v_j,

$$d(u_i, v_j) = \sum_{k=1}^{d} |v_j^k - u_i^k|.$$

Hence, there exists $\delta^{i,j} \in \Delta$ such that

$$d(v_i, v_j) = \sum_{k=1}^{d} \delta_k^{i,j}(v_j^k - u_i^k).$$

Next we define a collection L^d of $2^d n$ lines (planar equations) as follows: For $j = 1, \ldots, n$, consider the 2^d equations

$$\left\{ y = w_j \left[\left(\sum_{k=1}^{d} \delta_k v_j^k \right) - x \right] : \delta_k \in \{+1, -1\}, \right.$$
$$\left. k = 1, \ldots, d \right\}.$$

We uniquely identify each line in L^d by a pair $(j, \delta(\beta))$, where $j = 1, \ldots, n$, and $\delta(\beta) \in \Delta$.

As described above, using the incremental algorithm in [6,7], we first construct the arrangement of the collection L^d in $O((2^d n)^2)$ time.

For each $i = 1, \ldots, m$, generate the set of 2^d reals, A_i, defined by

$$A_i = \left\{ \sum_{k=1}^{d} \delta_k u_i^k : \delta_k \in \{+1, -1\}, \quad k = 1, \ldots, d \right\}.$$

Each $\alpha \in A_i$ is uniquely identified by some $\delta(\alpha) \in \Delta$.

Next for each $i = 1, \ldots, m$, (individually) apply the following step.

For each $\alpha \in A_i$, consider the vertical line $x = \alpha$ and apply (individually) an additional single stage of the incremental algorithm in [6,7], and obtain in $O(2^d n)$ time the sorted sequence of the y values of the intersection points of this line with the lines in L^d. Let C_i^α denote this sorted list (sequence).

Since $|A_i| = 2^d$ and $|C_i^\alpha| = O(2^d n)$ for $\alpha \in A_i$, in $O(d 2^{2d} n)$ time we can merge all the lists $\{C_i^\alpha\}$, $\alpha \in A_i$, ([1]). Let C_i denote the merged list.

Note that each value in the sorted list C_i is uniquely identified by a triplet $(j, \delta(\beta), \delta(\alpha))$. Therefore, using the above notation, for each $j = 1, \ldots, n$, the term $w_j d(v_j, u_i)$ is the unique value of C_i identified by the triplet $(j, \delta^{i,j}, \delta^{i,j})$. Thus, we conclude that the real vector $D^*(u_i)$ is exactly the subsequence of C_i corresponding to the n identifiers $(j, \delta^{i,j}, \delta^{i,j})$, $j = 1, \ldots, n$. The total time needed to construct $D^*(u_i)$ is therefore $O(d 2^{2d} n)$.

The total time to construct all the vectors $\{D^*(u_i)\}$, $i = 1, \ldots, m$, is therefore $O((2^d n)^2 + d 2^{2d} mn)$.

Theorem 2.1. *It takes* $O((2^d n)^2 + d 2^{2d} mn)$ *time to generate all the vectors* $D^*(u_i)$, $i = 1, \ldots, m$, *for the rectilinear problem in* R^d.

The above results on the rectilinear case can easily be extended to the l_∞ norm in R^d, where

$$d(u_i, v_j) = \max\{|v_j^k - u_i^k| : k = 1, \ldots, d\}.$$

In fact, in this case the d-dimensional problem reduces to d one-dimensional problems. Hence, we have the following result.

Theorem 2.2. *It takes* $O(dn^2 + dmn)$ *time to generate all the vectors* $D^*(u_i)$, $i = 1, \ldots, m$, *for the* l_∞ *problem in* R^d.

3. Sorting in tree network spaces

In this section we extend the above results to tree network spaces.

Given is an undirected tree graph $T = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the node set and E is the edge set. Each edge $e \in E$ has a nonnegative edge length, l_e. The edge lengths induce a distance function on T. For any pair of nodes, $v_i, v_j \in V$ we let $d(v_i, v_j)$ denote the length of the unique simple path $P(v_i, v_j)$, connecting v_i and v_j. In this section we consider only the weighted case, and therefore we assume that $S = \{u_1, \ldots, u_m\}$ coincides with V. (If $S \neq V$ we can augment the points in S to V, assign a zero weight to each point in $S - V$, and replace n by $n + m$.)

We use a recursive approach which is based on a centroid decomposition of the tree [9,23], to compute the vectors $D^*(v_i)$, $i = 1, \ldots, n$.

A centroid of the tree is a node v characterized by the property that each of the connected components obtained by the removal of v, contains at most $\frac{n}{2}$ nodes. A centroid can be found in linear time, and it is also an unweighted 1-median of the tree, [16,17]. Moreover, the tree can be decomposed into two subtrees, say, T^1 and T^2 with respective node sets V^1 and V^2 such that, $V^1 \cup V^2 = V$, $V^1 \cap V^2 = \{v\}$, $|V^1| \leqslant (2n+1)/3$, and $|V^2| \leqslant (2n+1)/3$.

We now describe the recursive approach.

Find a centroid v of the tree.

Recursively, for each $v_i \in V^1$, compute L_i^1, the sorted list of weighted distances of all nodes $v_j \in V^1$ from v_i. (L_i^1 is the sorted list of the elements in the multi-set $A_i^1 = \{w_j d(v_j, v_i) : v_j \in V^1\}$.) Similarly, for each $v_i \in V^2$, compute L_i^2, the sorted list of weighted distances of all nodes $v_j \in V^2$ from v_i. (L_i^2 is the sorted list of the elements in the multi-set $A_i^2 = \{w_j d(v_j, v_i) : v_j \in V^2\}$.)

For each $v_i \in V^1$, define K_i^1 to be the sorted list of the elements in the multi-set $B_i^1 = \{w_j d(v_j, v_i) : v_j \in V^2 - \{v\}\}$. Similarly, for each $v_i \in V^2$, define K_i^2 to be the sorted list of the elements in the multi-set $B_i^2 = \{w_j d(v_j, v_i) : v_j \in V^1 - \{v\}\}$. It then follows from these definitions that for each

$v_i \in V^1$, $(v_i \in V^2)$, $D(v_i)=A_i^1 \cup B_i^1$, $(D(v_i)=A_i^2 \cup B_i^2)$. Therefore, for each $v_i \in V^1$, $(v_i \in V^2)$, the vector $D^*(v_i)$ is defined by the vector (sorted list) obtained by merging L_i^1 with K_i^1 (L_i^2 with K_i^2).

We now show how to generate the sorted lists K_i^1, $v_i \in V^1$, and K_i^2, $v_i \in V^2$. Due to symmetry we focus only on K_i^1.

For each $v_j \in V$, let $a_j = d(v_j, v)$. Therefore, K_i^1 is the sorted list of the elements in the multi-set $\{w_j(a_j+a_i) : v_j \in V^2 - \{v\}\}$. From the above results on the case of the real line we conclude that the total effort to generate all the lists K_i^1, $v_i \in V^1$, is $O(n^2)$ time.

At this stage we have all the lists K_i^1, L_i^1, $v_i \in V^1$, and K_i^2, L_i^2, $v_i \in V^2$. Therefore, for each $v_i \in V^1$ $(v_i \in V^2)$, we can obtain $D^*(v_i)$ in $O(n)$ time by applying a standard merging step, [1], to the sorted lists K_i^1 and L_i^1, (K_i^2 and L_i^2).

To evaluate the complexity of the above recursive procedure, let $T(n)$ denote the total effort needed to generate all the vectors $D^*(v_i)$, $v_i \in V$ in a tree with n nodes. From the above we obtain

$$T(n) \leqslant cn^2 + T(n_1) + T(n_2),$$

where c is a constant, $n_1 + n_2 = n + 1$, $n_1 \leqslant (2n+1)/3$ and $n_2 \leqslant (2n+1)/3$. Thus, we conclude that $T(n) = O(n^2)$.

Theorem 3.1. *It takes $O(n^2)$ time to generate all the vectors $D^*(v_i)$, $i = 1, \ldots, n$, for a tree network with n nodes.*

Remark 3.1. It is shown in [14] that the single facility discrete unweighted ordered-median problem on a tree network can be solved in $O(n^2)$ time. The last theorem implies that the weighted version can also be solved with the same complexity, since complete evaluation can be performed in $O(n^2)$ time.

4. Complete evaluation for special cases

We have shown above that in the cases of rectilinear spaces and tree networks, for an arbitrary objective function depending on the ordering of weighted distances to the server, a complete evaluation of the objective at all points in the feasible set S can be carried out in $O(n^2)$ time when $|S| = O(|V|)$. (We have assumed above that when $D^*(u_i)$ is given, it takes $O(n)$ time to evaluate the objective at $u_i \in S$.)

We note that in some special cases where the objective depends only on some components of $D^*(x)$, a complete evaluation of the objective at all feasible points in S can be performed in subquadratic time. Following are some examples.

Consider first the case of the weighted 1-center problem. The objective value depends only on the first component of $D^*(x)$, i.e., the maximum weighted distance to the server at x. It is easy to check that in the real line case of this model, where $V = \{v_1, \ldots, v_n\}$ and $S = \{u_1, \ldots, u_m\}$ are two sets of points on the real line, a complete evaluation of the objective at all points in V can be performed in $O(n \log n + m \log n)$ time. A minimum point over S can actually be obtained in $O(n+m)$ time by first solving the continuous version of the problem, where the server can be located anywhere on the line [21]. If x^* is the unique solution to the continuous version, then one of the two closest points in S on either side of x^* is a solution to the discrete 1-center problem. (Note that a complete evaluation for the discrete 1-center problem in R^d with the l_∞ norm, can be done in $O(dn \log n + dm \log n)$ time, since the model decomposes into d one-dimensional problems.)

Next consider the single facility discrete planar rectilinear k-centrum problem defined in the Introduction. Using the results in [2], it is shown in [14] that for the unweighted case, a complete evaluation can be performed in $O(n \log^2 n)$ time, when $S = V$. (Recall that by the unweighted model we refer to the case where $w_j = 1$, $j = 1, \ldots, n$.)

Another example where complete evaluation can be executed in subquadratic time is the discrete unweighted Euclidean planar 1-center problem. First, generate the furthest point Voronoi diagram of the points in V in $O(n \log n)$ time [25]. We can then evaluate the 1-center objective at any point of S in $O(\log n)$ time, by using point location [18,25]. The total time amounts to $O(n \log n + m \log n)$. (When $S = V$, the $O(n \log n)$ bound is actually optimal as shown in [20].)

Two other examples are the (weighted) discrete 1-median and 1-center problems on tree networks. Evaluating the median objective function at all nodes of the tree $T = (V, E)$ can be done in $O(n+m)$ time by a simple bottom-up algorithm. (See [26].) We next show that a complete evaluation of the weighted 1-center objective at all nodes of the tree can be executed in $O((n+m)\log(n+m))$ time.

4.1. Complete evaluation of the 1-center objective on a tree

Given the tree $T=(V,E)$ our goal is to evaluate the terms $c_i=\max\{w_j d(v_i,v_j) : v_j \in V\}$, for all $v_i \in V$. (As in Section 3, we assume that $S=V$.)

We apply the centroid decomposition of a tree introduced above.

Let v be a centroid of T, and let V^1, V^2 be the respective partition of V. (See Section 3.)

Recursively, for each $v_i \in V^1$ compute $a_i^1 = \max\{w_j d(v_i,v_j) : v_j \in V^1\}$, and for each $v_i \in V^2$ compute $a_i^2=\max\{w_j d(v_i,v_j) : v_j \in V^2\}$.

For each $v_i \in V^1$, define $b_i^1 = \max\{w_j d(v_i,v_j) : v_j \in V^2\}$, and for each $v_i \in V^2$, define $b_i^2=\max\{w_j d(v_i, v_j) : v_j \in V^1\}$.

Hence, for each $v_i \in V^1$, $c_i=\max\{a_i^1, b_i^1\}$, and for each $v_i \in V^2$, $c_i=\max\{a_i^2,b_i^2\}$. We show how to generate b_i^1, $v_i \in V^1$ and b_i^2, $v_i \in V^2$. Due to symmetry we focus only on the computation of b_i^1, $v_i \in V^1$.

We first use the approach in [23], which yields an $O(n\log^2 n)$ complexity, and then improve it to $O(n\log n)$ by applying the machinery in [9].

In $O(n)$ time compute all the distances $d(v,v_i)$, $v_i \in V$. With each $v_j \in V^2$, associate the single variable linear function $h_j(t)=w_j(d(v_j,v)+t)$. Next, define the pointwise maximum function,

$$F(t)=\max\{h_j(t) : v_j \in V^2\}.$$

$F(t)$ is a piecewise linear and convex function with at most $|V^2|$ breakpoints. Using a standard divide and conquer algorithm we generate the sequence of its breakpoints, and the corresponding values of $F(t)$ in $O(n\log n)$ time. With such a representation of $F(t)$ it takes $O(\log n)$ time to compute $F(t)$ for any value of t. Noting that $b_i^1=F(d(v_i,v))$, for $v_i \in V^1$, we conclude that it takes $O(n\log n)$ time to compute all the terms b_i^1, $v_i \in V^1$. Therefore, the total time spent at this stage of the algorithm is $O(n\log n)$.

To evaluate the complexity of the above recursive procedure, let $T(n)$ denote the total effort needed to generate all the terms c_i, $v_i \in V$ in a tree with n nodes. From the above we obtain

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

where c is a constant, $n_1+n_2=n+1$, $n_1 \leqslant (2n+1)/3$ and $n_2 \leqslant (2n+1)/3$. Thus, $T(n)=O(n\log^2 n)$.

The above procedure can be expedited to reduce the complexity to $O(n\log n)$. We apply the machinery in [9]. First, we note that at each stage of the recursion we can sort all the distances $d(v_i,v)$, $v_i \in V$, from the centroid v in $O(n)$ time, using sorted lists generated at earlier stages. (See [9].) Let L^1 denote the sorted list of elements in the multi-set $\{d(v_j,v) : v_j \in V^1\}$, and let L_-^2 denote the sorted list of elements in the multi-set $\{-d(v_j,v) : v_j \in V^2\}$.

Next consider the construction of the function $F(t)$. Note that for $v_j \in V^2$, $-d(v_j,v)$ is the unique zero of the linear function $h_j(t)$. Hence, L_-^2 is the sorted list of these $|V^2|$ zeroes. Using L_-^2 we can now construct the sequence of breakpoints of the function $F(t)$ in $O(n)$ time [25]. (Note that the latter construction is equivalent to the problem of finding the convex hull of a set of points in the plane, when the points are already sorted with respect to one of the coordinates.)

Finally, given the sequence of breakpoints of $F(t)$, and the list L^1, we can evaluate $b_i^1=F(d(v_i,v))$ for all the nodes $v_i \in V^1$ in $O(n)$ time.

To conclude, the recursive equation corresponding to the modified version of the algorithm is

$$T(n) \leqslant cn + T(n_1)+T(n_2).$$

Therefore, $T(n)=O(n\log n)$.

Theorem 4.1. *It takes* $O(n\log n)$ *time to compute the* 1-*center objective values at all the nodes of a tree network with n nodes.*

Remark 4.1. We have described a superlinear algorithm to evaluate the weighted 1-center objective at all nodes of a tree. We should also note that the minimum value of this objective can be found in linear time by the algorithm in [21].

Remark 4.2. We note that the $O(n\log n)$ algorithm from the last theorem can be replaced by a simple $O(n)$ algorithm when the 1-center problem is unweighted. Specifically, using the $O(n)$ algorithm in [12], we first find a diameter of the tree. Suppose without loss of generality that the nodes v_1, v_2 are the leaves of a diameter, i.e., $d(v_1,v_2)=\max\{d(v_i,v_j) : i,j=1,\dots,n\}$. Then it follows from the discussion in [12] that for each node v_t, the unweighted 1-center objective value at v_t is given by $\max\{d(v_t,v_1),d(v_t,v_2)\}$.

In the case of a path network the $O(n\log n)$ bound stated in the above theorem can be improved to $O(n)$. Suppose without loss of generality that the nodes of the path are points on the real line satisfying $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_n$. For $j=1,\ldots,n$, define $h_j^+(t)=w_j(t-v_j)$ and $h_j^-(t)=-w_j(t-v_j)$. Let

$$F(t)=\max\{\max\{h_j^+(t),h_j^-(t)\}:j=1,\ldots,n\}.$$

$F(t)$ is a piecewise linear convex function. By the arguments used above, the sequence of breakpoints of $F(t)$ can be constructed in $O(n)$ time. Moreover, given that the points satisfy $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_n$, the values of the weighted 1-center objective, $F(v_1),\ldots,F(v_n)$, can be obtained in $O(n)$ total time.

Theorem 4.2. *It takes $O(n)$ time to compute the 1-center objective values at all the nodes of a path network with n nodes.*

4.2. Complete evaluation of the k-centrum objective on a tree

The algorithmic results of the previous subsection can be extended to the k-centrum objective. Consider first the case of a path graph, and suppose that the nodes are points on the real line satisfying $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_n$. Using the above notation note that for each point x on the real line, the k-centrum objective, denoted by $H_k(x)$, is given by the sum of the k largest elements in the multi-set $\{h_j^+(x):j=1,\ldots,n\}\cup\{h_j^-(x):j=1,\ldots,n\}$.

The function $H_k(x)$ is piecewise linear and convex, and its minimum value can be computed in $O(n)$ time [24]. Moreover, using the duality between points and lines in the plane, from Theorem 3.3 in [5] we conclude that the number of breakpoints of $H_k(x)$ is $O(nk^{1/3})$. (Actually, the exact number can be much smaller. $O(nk^{1/3})$ is an upper bound on the number of breakpoints of the k-level function, whose set of breakpoints is a super set of the set of breakpoints of $H_k(x)$. For example, in the unweighted case, all the slopes of $H_k(x)$ are integers bounded between $-k$ and $+k$, and therefore, due to convexity, it has at most $2k$ breakpoints.)

Algorithms to generate all the above breakpoints (and the respective values of $H_k(x)$) are reported in [3,8,13]. The one with the best complexity is the (randomized expected time) $O(nk^{1/3}\alpha(nk^{1/3})\log n)$ algorithm in [13]. (For each integer m, $\alpha(m)$ is the inverse of the Ackermann function, [27].) The $O(nk^{1/3}\log^2 n)$ algorithms in [3,8] are deterministic.

To conclude, we have the following result.

Theorem 4.3. *It takes $O(nk^{1/3}\log^2 n)$ time to compute the k-centrum objective values at all the nodes of a path network with n nodes.*

Turning to tree networks we can combine the last result with the centroid decomposition to obtain a subquadratic algorithm for complete evaluation of the k-centrum objective on a tree network.

We use the above notation. Recursively, for each node $v_i\in V$, we will maintain the set of the k largest elements in the multi-set $D(v_i)=\{w_jd(v_i,v_j):v_j\in V\}$. Let (V^1,V^2) be a centroid decomposition of the tree $T=(V,E)$ mentioned in the previous subsection. Define $k_1=\min\{k,|V^1|\}$, $k_2=\min\{k,|V^2|\}$, $k_1'=\min\{k,|V^1|-1\}$ and $k_2'=\min\{k,|V^2|-1\}$.

For each node $v_i\in V^1$, $(v_i\in V^2)$, let C_i^1, (C_i^2), be the subset of the k_1, (k_2), largest elements in $\{w_jd(v_i,v_j):v_j\in V^1\}$, $(\{w_jd(v_i,v_j):v_j\in V^2\})$.

For each node $v_i\in V^1$, $(v_i\in V^2)$, let D_i^1, (D_i^2), be the subset of the k_2', (k_1'), largest elements in $\{w_jd(v_i,v_j) : v_j\in V^2-\{v\}\}$, $(\{w_jd(v_i,v_j) : v_j\in V^1-\{v\}\})$.

Note that for each $v_i\in V^1$, $(v_i\in V^2)$, the set of the k largest elements in $D(v_i)$ is the set of the k largest elements in the multi-set $C_i^1\cup D_i^1$, $(C_i^2\cup D_i^2)$. (Since $|C_i^1\cup D_i^1|\leqslant 2k$, $(|C_i^2\cup D_i^2|\leqslant 2k)$, it takes $O(k)$ time to compute the k largest elements of $D(v_i)$, when the sets C_i^1,D_i^1, (C_i^2,D_i^2), are given, see [1].)

Due to symmetry we focus only on the construction of the sets D_i^1, $v_i\in V^1$.

Let $H_k^2(t)$ be the function which is the sum of the largest k_2' functions in the collection of linear functions $\{w_j(d(v,v_j)+t):v_j\in V^2-\{v\}\}$. From the above discussion it takes $O(nk^{1/3}\log^2 n)$ time to generate the entire sequence of breakpoints of $H_k^2(t)$. (We note that for all values of t, lying between two consecutive breakpoints, the set of k_2' largest functions is fixed.)

Next, for each $v_i\in V^1$, the set D_i^1 corresponds to the set of k_2' largest functions defining the value

$H_k^2(d(v,v_i))$. Hence, assuming that the distances $\{d(v,v_i)\}$, $v_i \in V^1$, are already sorted, by scanning the sequence of breakpoints of $H_k^2(t)$, we construct all the sets D_i^1, $v_i \in V^1$, in $O(kn)$ time.

To evaluate the total effort spent by this recursive algorithm, we denote by $T(n)$ the total time to compute the k largest elements in $D(v_i)$, for all $i = 1,\ldots,n$. We have the following recursion:

$$T(n) \leqslant c(nk^{1/3}\log^2 n + kn) + T(n_1) + T(n_2),$$

where c is a constant, $n_1 + n_2 = n + 1$, $n_1 \leqslant (2n+1)/3$ and $n_2 \leqslant (2n+1)/3$. Thus, we conclude with the next result.

Theorem 4.4. *It takes* $O(nk^{1/3}\log^3 n + kn\log n)$ *time to compute the k-centrum objective values at all the nodes of a tree network with n nodes.*

Remark 4.3. We note in passing that the time algorithm $O(nk^{1/3}\log^3 n + kn\log n)$ to compute the k-centrum objective from the last theorem, can be replaced by an $O(n\log^2 n)$ algorithm when the k-centrum problem is unweighted. We can direcly use the data structure in [23] which provides a compact $O(n\log n)$ representation of the set of unweighted distances between all pairs of nodes of the tree. With this data structure, for each node v_i it takes $O(\log^2 n)$ time to compute the k-centrum objective at v_i. Therefore, a complete evaluation can be executed in $O(n\log^2 n)$ time.

5. Related open problems

We have presented above some improved algorithms to evaluate objective functions corresponding to several discrete single facility location problems. In some important cases the $O(n^2)$ procedure for a complete evaluation, (in the case when $|S| = O(|V|)$), is the fastest known procedure to find the minimum value of the objective over the discrete feasible set. The question is whether better complexity bounds are attainable for finding the minimum in these cases. We list some examples.

Consider first the discrete rectilinear weighted 1-center and 1-median problems in R^d, $d \geqslant 2$. The median problem decomposes into d one-dimensional problems, and, therefore, can be solved in $O(dn\log n)$ time [19]. It is not known how to find the minimum solution to the discrete center problem in subquadratic time for a fixed $d > 3$. (For $d = 2$, the rectilinear (l_1 norm) model is equivalent to the l_∞ norm problem, and therefore, as noted above, a complete evaluation can be performed in $O(n\log n)$ time.) For comparison purposes note that the continuous version is solvable in $O(n)$ time [22,24]. The same results and questions apply to the k-centrum objective defined in the Introduction. The continuous version in R^d is solvable in $O(n)$ time in [24], but no subquadratic algorithm is known for the discrete version for $d \geqslant 2$. (Of course, the above $O(n^2)$ complete evaluation scheme identifies the minimum value.) More generally, consider the case of the ordered-median objective, where $\lambda_1 \geqslant \lambda_2 \cdots \geqslant \lambda_n \geqslant 0$. In this case the objective is piecewise linear and convex. Subquadratic algorithms for the continuous case are presented in [14]. For this objective, it is not known whether complete evaluation can be executed in subquadratic time even for path graphs.

Finally, we mention the Euclidean case. To the best of our knowledge, it is not known how to generate the sets $D^*(v_i)$, $v_i \in V$, in $O(n^2)$ time, even for the unweighted planar Euclidean case.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.
[2] S. Bespamyatnikh, K. Kedem, M. Segal, A. Tamir, Optimal facility location under various distance functions, Internat. J. Comput. Geom. Appl. 10 (2000) 523–534.
[3] R. Cole, M. Sharir, C. Yap, On K-hulls and related problems, SIAM J. Comput. 16 (1987) 61–77.
[4] J. Current, H. Min, D. Schilling, Multiobjective analysis of facility location decisions, European J. Oper. Res. 49 (1990) 295–307.
[5] T. Dey, Improved bounds on planar K-sets and related problems, Discrete Comput. Geom. 19 (1998) 373–383.
[6] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer, New York, Berlin, Heidelberg, 1987.
[7] H. Edelsbrunner, J. O'Rourke, R. Seidel, Constructing arrangements of lines and hyperplanes, SIAM J. Comput. 15 (1986) 341–363.
[8] H. Edelsbrunner, E. Welzl, Constructing belts in two-dimensional arrangements with applications, SIAM J. Comput. 15 (1986) 271–284.
[9] G.N. Frederickson, D.B. Johnson, Finding the kth paths and p-centers by generating and searching good data structures, J. Algorithms 4 (1983) 61–80.

[10] N. Halman, A linear time algorithm for the weighted lexicographic rectilinear 1-center problem in the plane, Inform. Process. Lett. 86 (2003) 121–128.

[11] H.W. Hamacher, S. Nickel, Multicriteria planar location problems, European J. Oper. Res. 94 (1996) 66–86.

[12] G.Y. Handler, Minimax location of a facility in an undirected tree graph, Transportation Sci. 7 (1973) 287–293.

[13] S. Har-Peled, Taking a walk in a planar arrangement, SIAM J. Comput. 30 (2000) 1341–1367.

[14] J. Kalcsics, S. Nickel, J. Puerto, A. Tamir, Algorithmic results for ordered median problems defined on networks and the plane, Oper. Res. Lett. 30 (2002) 149–158.

[15] S. Kapoor, Dynamic maintenance of maxima of 2-D point sets, SIAM J. Comput. 29 (2000) 1858–1877.

[16] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems: part 1. The p-centers, SIAM J. Appl. Math. 37 (1979) 513–538.

[17] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems: part 2. The p-medians, SIAM J. Appl. Math. 37 (1979) 539–560.

[18] D.G. Kirkpatrick, Optimal search in planar subdivisions, SIAM J. Comput. 12 (1983) 28–35.

[19] Y. Konforty, A. Tamir, The rectilinear single facility location problem with minimum distance constraints, Location Sci. 5 (1998) 147–163.

[20] D.T. Lee, Y.F. Wu, Geometric complexity of some location problems, Algorithmica 1 (1986) 193–212.

[21] N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, SIAM J. Comput. 12 (1983) 759–776.

[22] N. Megiddo, Linear programming in linear time when the dimension is fixed, J. Assoc. Comput. Mach. 31 (1984) 114–127.

[23] N. Megiddo, A. Tamir, E. Zemel, R. Chandrasekarn, 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.

[24] W. Ogryczak, A. Tamir, Minimizing the sum of the k-largest functions in linear time, Inform. Process. Lett. 85 (2003) 117–122.

[25] F. Preparata, M. Shamos, Computational Geometry: An Introduction, Springer, New York, Berlin, Heidelberg, 1985.

[26] A. Rosenthal, J.A. Pino, A generalized algorithm for centrality problems on trees, J. Assoc. Comput. Mach. 36 (1989) 349–361.

[27] M. Sharir, P.K. Agarwal, Davenport-Schinzel Sequences and Their Geometric Applications, Cambridge University Press, Cambridge, 1995.

[28] R.E. Steuer, Multiple Criteria Optimization: Theory, Computation and Applications, Wiley, New York, 1986.

[29] A. Tamir, An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs, Oper. Res. Lett. 19 (1996) 59–64.

[30] A. Tamir, The k-centrum multi-facility location problem, Discrete Appl. Math. 109 (2001) 293–307.