ELSEVIER

Discrete Applied Mathematics 109 (2001) 293–307

DISCRETE APPLIED MATHEMATICS

The k-centrum multi-facility location problem

Arie Tamir[1]

Department of Statistics and Operations Research, School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel

Received 17 February 1999; revised 23 December 1999; accepted 30 May 2000

Abstract

The most common problems studied in network location theory are the p-center and the p-median problems. In the p-center problem the objective is to locate p service facilities to minimize the maximum of the service distances of the n customers to their respective nearest service facility, and in the p-median model the objective is to minimize the sum of these n service distances. (A customer is served only by the closest facility.) We study the p-facility k-centrum model that generalizes and unifies the above problems. The objective of this unifying model is to minimize the sum of the k largest service distances. The p-center and the p-median problems correspond to the cases where $k = 1$ and n, respectively. We present polynomial time algorithms for solving the p-facility k-centrum problem on path and tree graphs. These algorithms can be combined with the general approximation algorithms of Bartal (Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 161–168) and Charikar et al. (Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 114–123) to obtain an $O(\log n \log\log n)$ approximation for a p-facility k-centrum problem defined on a general network.

Keywords: Center location problem; Median location problem; Centrum location problem; Tree graphs; Approximation algorithms

1. Introduction

Let $G = (V,E)$ be an undirected graph with a node set $V = \{v_1,\ldots,v_n\}$ and an edge set E. Each edge has a nonnegative length and is assumed to be rectifiable. Thus, we will refer to interior points on an edge by their distances (along the edge) from the two nodes of the edge. We let $A(G)$ denote the continuum set of points on the edges of G. The edge lengths induce a distance function on $A(G)$. For any x, y in $A(G)$,

[1] This research is sponsored in part by the National Sciene Foundation, Grant DMI-9522882.
E-mail address: atamir@math.tau.ac.il (A. Tamir)

0166-218X/01/$ - see front matter
PII: S0166-218X(00)00253-5

$d(x, y)$ will denote the length of a shortest path connecting x and y. Also, if Y is a subset of $A(G)$, we define the distance of x from Y by

$$d(x, Y) = d(Y, x) = \inf\{d(x, y): y \in Y\}.$$

In this study we consider the following p-facility k-centrum location problem on the graph G. This problem generalizes and unifies the classical p-median and p-center problems.

In the p-facility k-centrum problem on $G = (V, E)$ we wish to find a subset S of p points (service facilities or centers) in $A(G)$, minimizing the sum of the k farthest (w.r.t. weighted distances), nodes (customers) from S. Specifically, suppose that each node v_i, $i = 1, \ldots, n$, is associated with a nonnegative weight w_i. For each vector $(x_1, \ldots, x_n) \in R^n$, define $H_k^n(x_1, \ldots, x_n)$ to be the sum of the k largest entries in the set $\{x_1, \ldots, x_n\}$. The objective function of the p-facility k-centrum problem is to find a subset S of $A(G)$, $|S| = p$, minimizing

$$F(S) = H_k^n(w_1 d(S, v_1), \ldots, w_n d(S, v_n)).$$

In the discrete version of the p-facility k-centrum problem we also require S to be a subset of nodes. A minimizer of $F(S)$ is called an *absolute k-centrum*, and an optimal solution to the discrete problem is a *discrete k-centrum*. The unweighted p-facility k-centrum problem refers to the case where $w_i = 1$, $i = 1, \ldots, n$.

From the definition it follows that if $k = 1$ the above model reduces to the classical p-center problem, and when $k = n$ we obtain the classical p-median problem. Several special cases of the single facility k-centrum problem have been discussed in the literature. To the best of our knowledge, the k-centrum concept was first introduced by Slater [23]. Slater [23] and Andreatta and Mason [2,3], considered the case of the discrete single facility on tree graphs, and observe some properties including the convexity of the objective on tree graphs. There are no algorithms with specified complexity in these papers. Peeters [21] studied the single facility problem on a graph, which he called the upper-k 1-median. He presented an $O(n^2 \log n + |E|n)$ algorithm for solving only the discrete version of this model, where the centrum is restricted to be a node. We are not aware of any studies in the literature on the multiple facility k-centrum model.

As a generalization of the p-center and the p-median problems, the p-facility k-centrum problem is also NP-hard on general networks, Kariv and Hakimi [17]. In Section 2 we present polynomial time algorithms for finding a single facility k-centrum on path graphs, tree graphs and general graphs. In Section 3 we describe polynomial time, dynamic programming algorithms for solving the multi-facility k-centrum problems on path and tree graphs. These algorithms can be combined with the general approximation algorithms of Bartal [4] and Charikar et al. [6] to obtain $O(\log n \log\log n)$ approximation for a p-facility k-centrum problem defined on a general network.

2. Single facility k-centrum

2.1. The single k-centrum of a path

We consider first the case of a single facility ($p = 1$) on a line (a path graph). In this case the subset S in the definition of the model is viewed as a point x on the real line, and each node v_i is also regarded as a real point. Thus,

$$F(x) = H_k^n(w_1|x - v_1|, \ldots, w_n|x - v_n|).$$

To compute $F(x)$ in the case of the single facility k-centrum problem on the line we look at the k largest functions in the collection $\{w_i|x-v_i| = \max(w_i(x-v_i), -w_i(x-v_i))\}$, $i=1,\ldots,n$. Since $k \leqslant n$, we may assume that for each i, and a real x, at most one of the 2 values, $\{w_i(x-v_i), -w_i(x-v_i)\}$, i.e., a nonnegative one, belongs to the subcollection of the k largest function values at x. Thus, the single facility k-centrum problem on the line amounts to finding the minimum point of the sum of the k largest linear functions in the collection consisting of the above $2n$ linear functions, $\{w_i(x - v_i), -w_i(x - v_i)\}$, $i = 1,\ldots,n$.

To solve the above problem we consider the following more general problem. Let $[a,b]$ be an interval on the real line, and let k be a positive integer. Given is a collection of $m \geqslant k$ linear functions $\{a_i x + b_i\}$, $i = 1,\ldots,m$, defined on $[a,b]$. The objective is to find x^*, a minimum point of $F(x) = H_k^m(a_1x + b_1, \ldots, a_mx + b_m)$ in $[a,b]$.

It is known that if we have a set of m linear functions, the sum of the k largest functions is a convex, piecewise linear function. Dey [10] showed that the number of breakpoints of this convex function is bounded above by $O(mk^{1/3})$. Katoh (see [12]), demonstrated that the number of breakpoints can achieve the bound $\Omega(m \log k)$.

To find the optimum, x^*, we use the general parametric procedure of Megiddo [19], with the modification in Cole [9]. 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 m linear functions, $\{a_i x + b_i\}$, $i = 1,\ldots,m$ (using x as the parameter). The test for determining the location of a given point x' w.r.t. x^* is based on the linear time algorithm for finding the kth largest element of a set of reals [5]. Due to the convexity of the function $F(x)$, it is sufficient to determine the signs of the one-sided derivatives at x'. This is done as follows. Consider, for example, $F^+(x')$, the derivative from the right. First, using the algorithm in [5] we compute in linear time, the function value $F(x')$, and the kth largest linear function at x'. If there is only one function in the collection of the m functions whose value is equal to the kth largest value at x', then $F^+(x')$ is the sum of the slopes of the k linear functions corresponding to the k largest values at x'. Otherwise, let $k' < k$, be the number of linear functions whose value at x' is larger than the kth largest value, and let A be the sum of the slopes of these k' functions. Let I, be the subcollection of linear functions whose value at x' is equal to the kth largest value. Again, using the algorithm in [5] we compute the $k - k'$ largest slopes from the set of slopes of the linear functions in I. Let B be the sum of these $k - k'$ slopes. We then have $F^+(x') = A + B$. We

now conclude that with the above test the parametric approach in [19,9] will find x^* in $O(m \log m)$ time. In the k-centrum problem on the line $m = 2n$ and therefore the complexity to compute the absolute k-centrum is $O(n \log n)$.

Because of convexity the solution to the discrete k-centrum problem (the discrete k-centrum) is one of the two consecutive nodes bounding the absolute k-centrum. The discrete solution can also be found directly by a simpler algorithm. First sort the nodes, and then, using the above linear time procedure to compute the derivatives of $F(x)$, perform a binary search on the nodes. This approach also takes $O(n \log n)$ time.

In the unweighted case ($w_i = 1$, for $i = 1, \ldots, n$), the k-centrum can be found in linear time. Define $l = [(k+1)/2]$. (For any real number y, $[y]$ denotes the smallest integer which is greater than or equal to y.) It is easy to verify that the midpoint of the interval connecting the lth smallest point in V with the lth largest point in V is a local minimum point. By the convexity of the objective function, we conclude that it is an absolute k-centrum. Thus, by using the linear time algorithm, in [5], we can find the unweighted k-centrum of an unsorted sequence of n points on the line in $O(n)$ time.

Conceptually, the above characterization can be extended to the weighted case. But it is not clear whether this can be used to obtain a linear time algorithm for the weighted case. A necessary and sufficient condition in the weighted case is that the k farthest (w.r.t. weighted distances) nodes will be evenly located on both sides of the centrum, i.e., the weight of those nodes on either side of the centrum should be half of the total weight of the k nodes. In the unweighted case the total weight is fixed, and is equal to k, while in the weighted case, the total weight of the k furthest nodes is not known apriori.

At this point it is not clear whether the weighted problem can be solved in linear time. However, we next present an algorithm which solves the weighted model for a fixed value of k, in $O(n)$ time. Specifically, for any fixed value of k, this algorithm will find the (weighted) k-centrum x^* on the line in $O((k^2 \log k)n)$ time.

Again, we consider a general collection of m linear functions $\{a_i x + b_i\}$, $i = 1, \ldots, m$. Let x^* be a minimum point of the function $F(x)$, defined above, in $[a, b]$. We will show that in $O(m)$ time we can remove a fixed proportion of the linear functions from the entire collection, without affecting the optimality of x^*.

First divide the collection of m functions into $m/(k+1)$ sets of $k+1$ functions each. (For simplicity suppose that $m/(k+1)$ is integer.) For each set of $k+1$ functions, at most k of the functions will appear in the objective. Specifically, for each x, the mimimum of the $k+1$ functions in the set is ineffective, and therefore can be dropped. The minimum of the $k+1$ functions of a set, called the minimum envelope, is a concave piecewise linear function with k breakpoints. Because of the convexity of the objective, if we know the pair of breakpoints bracketing x^*, we can then omit the minimum function (on the interval connecting the pair), from this collection of $k+1$ linear functions. We proceed as follows. For each one of the above $m/(k+1)$ sets of $k+1$ functions, we find in $O(k \log k)$ time its minimum envelope [18]. The total effort of this step is $O(m \log k)$. Next we look at the set P_1, consisting of the $m/(k+1)$ median breakpoints

of the $m/(k+1)$ envelopes. We find the median point in P_1, and check its position w.r.t. x^* in $O(m)$ time, as explained above, by implementing the algorithm in [5]. Note that the time to check the location of a given point with respect to x^* is proportional to the number of linear functions in the (current) collection. Thus, for at least half of the $m/(k+1)$ envelopes we know the position of their medians w.r.t. the centrum. We consider only these $[m/2(k+1)]$ envelopes and continue this binary search. For example, in the next iteration each such envelope contributes its $[k/4]$th or $[3k/4]$th breakpoint, depending on the location of the centrum x^* w.r.t. its median. Let P_2 be the set of these $[m/2(k+1)]$ points. We now check the position of the median element in P_2, w.r.t. x^*. After $O(\log k)$ iterations we identify $O(m/(k(k+1)))$ sets (envelopes) such that for each one of them we have a pair of adjacent breakpoints (segment) bracketing x^*. Thus, from each one of these $O(m/(k(k+1)))$ envelpoes, we can eliminate the linear function which coincides with the envelope on this segment. To summarize, at the end of the first phase, in a total effort of $O(m \log k)$ ($O(\log k)$ iterations of $O(m)$ time each), we eliminate $O(m/k^2)$ functions. We then recursively restart the process with the remaining subcollection of linear functions. Again, it should be emphasized that the time to check the location of a given point with respect to the centrum is proportional to the number of linear functions in the remaining subcollection of linear functions. In particular, the total effrot spent in the second phase is $O(m(1-1/k^2)\log k)$. At the last phase the subcollection will consist of $O(k^2)$ functions, and therefore, from the discussion above the time needed to find x^* for this subcollection is $O(k^2 \log k)$.

To evaluate the complexity of the above algorithm, suppose that we have a collection of m linear functions. Let $T(m)$ be the total effort to compute x^*, a minimum point of the function, defined as the sum of the k largest functions in the collection. (In the single k centrum problem on a path $m = 2n$.)

The recursive equation is

$$T(m) \leqslant c_k m + T(a_k m),$$

where $c_k = c \log k$, for some constant c, and $a_k = 1 - 1/(k+1)^2$. Thus, $T(m) = O((k^2 \log k)m)$.

We note in passing that with the same effort we can also find the discrete k-centrum, since it is one of the pair of adjacent nodes bracketing the absolute k-centrum.

2.2. *The single k-centrum of a tree*

The above algorithm for a path graph can be extended to a tree $T = (V, E)$, $V = \{v_1, \ldots, v_n\}$. We use a 2-phase approach. In the first phase we identify a pair of adjacent nodes (an edge) such that the absolute k-centrum is between them. In the second phase we have to solve the above problem on the edge (segment) containing the absolute k-centrum. The latter subproblem is a special case of the problem on a path discussed in the previous section. Therefore, the total time for the second phase is $O(n \log n)$. We will next show that the first phase can be performed in $O(n \log^2 n)$ time.

From the discussion on path graphs we know that the objective function is convex on each path of the tree. Therefore, we can apply the centroid decomposition search procedure of Kariv and Hakimi [17]. They used this search procedure to find the weighted 1-center of a tree. The procedure is based on searching over a sequence of $O(\log n)$ nodes, which are centroids of some nested sequence of subtrees. (The total time needed to generate the sequence is only $O(n)$.) For each node in this sequence they have a simple $O(n)$ test to identify whether this node is the 1-center of the tree. Moreover, if the node is not the 1-center, the test identifies the direction from the node towards the 1-center. All we need now to implement their search procedure is an efficient test which finds a direction from a given node v to the absolute k-centrum if v itself is not the absolute k-centrum.

Consider an arbitrary node v of the given tree $T=(V,E)$. Let $T_1=(V_1,E_1),\ldots,T_q=(V_q,E_q)$ denote the set of all maximal connected components obtained by the removal of v and all its incident edges from T. For $i=1,\ldots,q$, let u_i be the closest node to v in $T_i=(V_i,E_i)$. Let r be the kth largest element in the set of weighted distances of the nodes of T from v. For $i=1,\ldots,q$, let V_i^+ ($V_i^=$) be the index set of the nodes in T_i, whose weighted distance from v is greater than (equal to) r. Let $t_i=|V_i^+|$, $s_i=|V_i^=|$, and let T_i^+ denote the subtree obtained by augmenting the edge (v,u_i) to T_i. Also, let $V^+=\bigcup_{i=1,\ldots,q} V_i^+$ and $V^=\bigcup_{i=1,\ldots,q} V_i^=$.

With the above notation we can state a simple test to determine whether the k-centrum is in a given T_j^+. Define $t=\sum_{i=1,\ldots q} t_i$, and $s=\sum_{i=1,\ldots q} s_i$. Note that $t\leqslant k\leqslant t+s$.

Consider a point $x\neq v$, on the edge (v,u_j), which is sufficiently close to v. Then it is easy to check that the difference between the objective value at x and the objective value at v is given by

$$d(v,x)(-A_j+B_j-C_j+D_j),$$

where $A_j=\sum_{h\in V_j^+} w_h$, $B_j=\sum_{h\in V^+-V_j^+} w_h$, D_j is the sum of the $m_j=\min(k-t,s-s_j)$ largest weights in the set $\{w_h\colon h\in V^=-V_j^=\}$, and C_j is the sum of the $k-t-m_j$ smallest weights in the set $\{w_h\colon h\in V_j^=\}$.

It is now clear that if v is not an absolute k-centrum, then the latter is contained in the component T_j^+, if and only if the above expression for the difference is negative. v is an absolute k-centrum if and only if $(-A_j+B_j-C_j+D_j)$ is nonnegative for all $j=1,\ldots,q$.

We next show that for a given node v, the total effort to compute the above dieference for all $j=1,\ldots,q$, is $O(n\log n)$. Combined with the centroid decomposition, which requires using this step $O(\log n)$ times, we will have an $O(n\log^2 n)$ algorithm to find the absolute k-centrum of a tree.

It is clear from the definitions that the total effort needed to compute A_j,B_j, for all $j=1,\ldots,q$, is $O(n)$. For each $j=1,\ldots,q$, C_j can be computed in $O(s_j)$ time by using the linear time algorithm in [5]. Thus, the total time to compute C_j for all $j=1,\ldots,q$, is $O(s)=O(n)$. The computation of the terms D_j, $j=1,\ldots,q$, is more involved.

We first sort the elements of the set $\{w_h\colon h\in V^=\}$ in decreasing order. Let $L=(w_{h(1)},w_{h(2)},\ldots,w_{h(s)})$ denote the sorted list. We then compute the partial sums

$W_p = \sum_{g=1,\ldots,p} w_{h(g)}$, $p = 1,\ldots,s$. The total effort of this step is $O(s \log s) = O(n \log n)$.

To compute a term D_j, $j = 1,\ldots,q$, we now perform the following steps. First we delete from the list L, the elements $\{w_h: h \in V_j^=\}$, and obtain the reduced sublist L_j. The effort needed is $O(s_j \log s)$. Next we identify the m_jth largest element in L_j. Suppose that this is the element $w_{h(y)}$ in the list L. The term D_j is now defined by $D_j = W_y - \sum\{w_{h(z)}: h(z) \in V_j^=, z < y\}$. The effort to compute D_j is $O(s_j \log s)$. Therefore, the total effort to compute all the terms D_j, $j = 1,\ldots,q$, is $O(s \log s) = O(n \log n)$.

To conclude, the above centroid decomposition search procedure will find an absolute k-centrum of a tree in $O(n \log^2 n)$ time. The discrete k-centrum is one of the 2 nodes of the edge containing the absolute k-centrum.

We note in passing that the time needed to find the k-centrum in the unweighted case is only $O(n + k \log n)$. First, observe that from the discussion in the previous section it follows that the time needed for the second phase is only $O(n)$. The first phase can be performed in $O(n + k \log n)$ time. There are $O(\log n)$ iterations in the first phase. In the first iteration we find the centroid of the original tree, say v. We compute the terms $\{A_j, B_j, C_j, D_j\}$, $j = 1,\ldots,q$, defined above. In the unweighted case $A_j = t_j$, $B_j = t - t_j$, $C_j = k - t - m_j$, and $D_j = m_j$, $j = 1,\ldots,q$. The effort of this step is therefore proportional to the number of nodes. If v is not the k-centrum, the k-centrum is in some subtree T_j^+. (Since v is a centroid the number of nodes in T_j is at most $n/2$.) Let V' be a subset of nodes in $V - V_j$, corresponding to a set of $\min([(k+1)/2], |V - V_j| - 1)$ largest distances from v to nodes in $V - V_j$. From the discussion on the unweighted k-centrum of a path in the previous section, we conclude that the unweighted k-centrum is determined only by its distances to the nodes of T_j^+ and the nodes in V'. Let T' denote the tree obtained by augmenting the node set V' to T_j^+, and connecting each node $v_t \in V'$ to the centroid v with an edge of length $d(v_t, v)$. The number of nodes of T' is at most $(k+1)/2 + 1 + n/2$. In the second iteration T' replaces the original tree T. We find the centroid of T', and proceed as above. Since there are $O(\log n)$ iterations, the total effort of the first phase is $O(n + k \log n)$. Thus, the time needed to locate a single k-centrum of a tree in the unweighted case is $O(n + k \log n)$. We believe that the $O(n \log^2 n)$ bound for the weighted case can be improved to $O(n \log n)$.

2.3. *The single k-centrum of a graph*

Peeters [21] studied the single facility problem on a general graph. He presented an $O(n^2 \log n + |E|n)$ algorithm for solving the discrete case only. Finding the absolute k-centrum is much more involved. We point out that this latter task can be performed efficiently by using modern computational geometry methods.

Like in the 1-center problem on a graph, the solution approach is to find the best local absolute k-centrum on each edge of the graph, and then select the global absolute k-centrum as the best of the $|E|$ local solutions.

In the preprocessing phase we compute the distances between all pairs of nodes. This can be done in $O(|E|n + n^2 \log n)$ time [13].

Consider an edge of the graph. The edge will be viewed as a line segment. In this case we have a collection of n "roof" functions, and we need the minimum point of the objective function defined as the sum of the k largest roof functions in the collection. (Each such roof function represents the weighted distance of a given node from the points on the edge, and it is the minimum of 2 linear functions, defined on the edge [18]). The objective function is piecewise linear, but it is not convex over the line segment representing the edge. A naive approach to find an optimal point on the segment is to compute the $O(n^2)$ intersections points of the $2n$ linear functions, and evaluate the objective function at each one of them. The effort will be superquadratic in n. We show how to improve upon this bound and obtain a subquadratic algorithm.

It is clear that a minimum value of this objective is attained at a breakpoint of the kth largest function of the given collection of n roof functions. Therefore, it will suffice to construct the piecewise linear kth largest function. (A piecewise linear function defined on the real line is represented by its sequence of breakpoints and their respective function values.)

Agarwal et al. [1] present results on the complexity bound of the kth largest function of a collection of "segment" functions. (A segment function is defined over the real line. It is linear over some segment of its domain, and is $-\infty$ elsewhere.) This bound (combined with Dey [10]) is $O(nk^{1/3}\alpha(n/k))$. (The function $\alpha(n/k)$ is the inverse Ackermann function, which is a very slow growing function of its argument. See [22].) This bound is useful in our context, since each roof function can be represented by two "segment" functions. The kth largest function of a collection of n segment functions can be constructed in $O(nk^{1/3}\alpha(n/k)\log^2 n)$ time by the algorithm in [11], which generates the kth largest function of a collection of linear functions. As noticed above, the local k-centrum on an edge is a breakpoint of the kth largest function of the collection of the n roof functions corresponding to that edge. Therefore, with this approach the total time to find the global single k-centrum will be $O(|E|nk^{1/3}\alpha(n/k)\log^2 n)$. (This effort dominates the preprocessing time.)

In the unweighted case the slopes of each roof function are $+1$ and -1, and the kth largest function of the collection of the n roof functions corresponding to a given edge, has $O(n)$ breakpoints only. (This follows from the fact that the increasing (decreasing) part of a roof function can coincide with the piecewise linear kth largest function on at most one interval connecting a pair of adjacent breakpoints of the kth largest function.) We note that a kth largest function of an edge can be constructed in $O(n \log n)$ time. For the sake of brevity we skip the details. The total time to find the global unweighted single k-centrum is therefore $O(|E|n \log n)$. (This effort dominates the preprocessing time.)

3. The multi-facility k-centrum

Since the p-facility k-centrum problem generalizes both the p-median and the p-center problems, it is NP-hard on general graphs. The recent papers by Charikar

et al. [8], Jain and Vazirani [16] and Charikar and Guha [7] provide constant-factor approximation algorithms for the p-median problem. At this point it is not clear whether the techniques in [8,16,7] can be modified to yield constant-factor algorithms for the p-facility k-centrum problem.

We next show that the problem is polynomially solvable on path and tree graphs. We note that polynomial algorithms for k-centrum problems on tree graphs can be used to approximate such problems on general graphs. The earlier papers by Bartal [4] and Charikar et al. [6] presented $O(\log p \log\log p)$ approximation algorithms for the p-median problem. These algorithms are based on solving p-median problems on a family of trees. (The general network metric is approximated by the tree metrics.) The basic approach yields an $O(\log n \log\log n)$ approximation algorithm for the p-median model. The same approach can be applied to the p-facility k-centrum model. Therefore, polynomial time algorithms for the latter model on trees are useful in deriving $O(\log n \log\log n)$ approximating solutions for k-centrum problems defined on general graphs. (The improved $O(\log p \log\log p)$ bound for the p-median problem, reported in [4,6] is derived by using a preprocessing stage to reduce the number of nodes to $O(p)$. It is not yet clear whether this preprocessing stage is also applicable for the p-facility k-median model.)

To simplify the presentation we first restrict our discussion to the discrete models where the facilities must be located at nodes of the graph.

Also, we assume that the problem is nondegenerate. Specifically, if $\{v_i, v_j\}$ and $\{v_s, v_t\}$ are two distinct pairs of nodes, then $d(v_i, v_j) \neq d(v_s, v_t)$ and $w_i d(v_i, v_j) \neq w_s d(v_s, v_t)$. We will later show how to resolve degenerate instances.

We augment a new parameter, r, to the above problem, and solve a modified problem for each relevant value of this parameter. We look at the *p-facility k-centrum problem with service distance r*, as the problem of minimizing the sum of the k largest weighted distances, provided the kth largest weighted distance is at least r, and the $(k+1)$th weighted distance is smaller than r. If there is no such solution the objective value will be regarded as ∞. Note that if $r = 0$, to get a finite value we must have $k = |V|$, and the problem reduces to the p-median problem. (An $O(pn)$ algorithm for the p-median problem on a path appears in [14]. $O(p^2n^2)$ and $O(pn^2)$ algorithms solving the p-median problem on a tree are given in [18] and [24], respectively.) Thus, we will assume that $r > 0$. Because of the nondegeneracy assumption, the solution to the original problem is the best solution to the parameterized version over all values of the parameter r.

3.1. The (discrete) p-facility k-centrum problem on a path

Following is a dynamic programming approach for the discrete case on a path.

We represent the path by its sorted set of n nodes $V = \{v_1, \ldots, v_n\}$, where v_{i+1} is adjacent to v_i, $i = 1, \ldots, n-1$. v_i, $i = 1, \ldots, n$, is also viewed as a real point, and thus, $v_i < v_{i+1}$, $i = 1, \ldots, n-1$.

For each pair $i < j$, a radius r, and an integer $q \leqslant k$, suppose that there are facilities at v_i and v_j only, and define $A(i,j,q)$ to be the sum of the q largest weighted distances of the nodes in the subpath $[v_i, v_j]$ to their respective nearest facility. Also, define $A'(i,j,q,r) = A(i,j,q)$ if the qth largest weighted distance is greater than or equal to r and the $(q+1)$th largest weighted distance is smaller than r, and $A'(i,j,q,r) = \infty$, otherwise. In particular, with the nondegeneracy assumption, for each r there is a unique value of q, $q(i,j,r)$, such that $A'(i,j,q,r)$ is finite.

For each i, a radius r, an integer $q \leqslant k$, and an integer $m \leqslant p$, define $G(i,q,r,m)$ to be the minimum of the sum of the q largest weighted distances of the nodes in the subpath $[v_i, v_n]$, to their respective nearest facilities, provided that there are m facilities in the interval (v_i is one of them), the qth largest weighted distance is at least r, and the $(q+1)$th largest weighted distance is smaller than r. Note that $m \leqslant p$, $m \leqslant n-i+1$, $q \leqslant k$, and $q \leqslant n-i+1$. If there is no such solution with the qth largest weighted distance being at least r, and the $(q+1)$th largest distance being smaller than r, define $G(i,q,r,m)=\infty$.

Suppose that r is a positive real number. We have the following recursion:

1. Let $i = n$. Then $G(n,0,r,m) = 0$ and $G(n,1,r,m) = \infty$.
2. Suppose $i < n$.
3. Let $m=1$. $G(i,q,r,1)$ is equal to the sum of the q largest weighted distances from v_i, if the qth largest weighted distance is at least r, and the $(q+1)$th largest weighted distance is less than r. Otherwise, $G(i,q,r,1) = \infty$.
4. Let $m \geqslant 2$. Then $G(i,q,r,m)$ is finite only if

$$G(i,q,r,m) = \min_{\{j:\ j>i, q(i,j,r)\leqslant q\}} \{A'(i,j,q(i,j,r),r) + G(j,q-q(i,j,r),r,m-1)\}.$$

We assume without loss of generality that an optimal solution to the problem on a path has a facility at v_1, otherwise, we can augment a node v_0, left of v_1, with $v_1 - v_0 = \infty$, and increase the number of facilities by 1.

From the above recursion it follows that the total effort to solve the discrete p-facility k-centrum problem with service radius r (i.e., compute $G(1,k,r,p)$), is $O(kpn^2)$. The parameter r can be restricted to values in the set $\{w_i|v_i - v_j|:\ i,j = 1,\ldots,n\}$. Therefore, assuming that all the terms $A(i,j,q)$ are already available, the complexity of the above scheme to solve the discrete p-facility k-centrum problem on a path is $O(kpn^4)$.

The preprocessing phase of computing the $A(i,j,q)$ functions is dominated by the above bound. For each pair $i < j$, there are only $O(n)$ distinct values of $A(i,j,q)$ corresponding to the values $r = w_t(v_t - v_i)$ or $r = w_t(v_j - v_t)$ for some $t = i, i+1,\ldots,j$. Hence, there are only $O(n^3)$ distinct values of all functions $A'(i,j,q,r)$.

For each pair $i < j$, the total time needed to compute the $O(n)$ values of $A(i,j,q)$ is $O(n \log n)$. Suppose, as above, that the nodes of the path are points on the real line with $v_i < v_{i+1}$, for $i = 1,\ldots,n-1$. We first sort the $O(n)$ elements in the set $\{\min\{w_t(v_t - v_i), w_t(v_j - v_t)\}:\ t = i, i+1,\ldots,j\}$. It then takes $O(n)$ additional time to compute $A(i,j,q)$, for all relevant values of q. Thus, the total preprocessing time is $O(n^3 \log n)$. In the unweighted case we can avoid the sorting and the preprocessing phase takes only $O(n^3)$ time.

3.2. The (discrete) p-facility k-centrum problem on a tree

Without loss of generality suppose that the given tree $T=(V,E)$ is binary and its root is v_1 (see [24]). For each node v_i in V, we let V_i denote the subset of all nodes in V that are *descendants* of v_i, i.e., v_i is on the paths connecting them to the root. In particular, for the root v_1, $V_1=V$. A *child* of a node v_i is a descendant of v_i, which is connected to v_i by an edge.

We now describe a recursive algorithm to solve the discrete p-facility k-centrum problem with service distance r, defined above. The algorithm is a "bottom-up" scheme which starts at the leaves of the rooted tree, and recursively computes solutions for subproblems corresponding to the subtrees defined by the sets V_i.

For each set $(v_i,q,r,m:v_j,v_k)$, where $v_j\in V_i$, $v_k\in V-V_i$, we let $G(v_i,q,r,m:v_j,v_k)$ be the minimum sum of the q largest weighted distances of the nodes in V_i to their respective nearest centers, provided there are m centers in V_i (with the closest to v_i being v_j), there is already at least one center at $V-V_i$, (the closest such center is v_k), the qth largest weighted distance is at least r, and the $(q+1)$th largest weighted distance is smaller than r. Note that $q\leqslant|V_i|$, $q\leqslant k$, $m\leqslant|V_i|$, and $m\leqslant p$. (If $m=0$, assume that v_j does not exist. By adding a superroot, say v_0, and connecting it to the original root by an edge of infinite length, we may assume that there is a center at v_0 and therefore v_k always exists.)

If there is no solution satisfying the requirements we set the value of the function G to be ∞. If $q=0$, then $G(v_i,q,r,m:v_j,v_k)$ is finite (and is equal to 0) only if there is a feasible solution with m centers such that the service distances of all nodes in V_i are less than r.

To simplify the recursive equations, for each pair of reals, x,r, define $a(x,r)=1$ if $x\geqslant r$, and $a(x,r)=0$, otherwise.

We now define the function $G(v_i,q,r,m:v_j,v_k)$ recursively.

1. Let v_i be a leaf of the rooted tree. Suppose that $m=0$. If $w_id(v_i,v_k)\geqslant r$, set $G(v_i,1,r,0:-,v_k)=w_id(v_i,v_k)$, and $G(v_i,0,r,0:-,v_k)=\infty$. Otherwise, set $G(v_i,1,r,0:-,v_k)=\infty$, and $G(v_i,0,r,0:-,v_k)=0$. Suppose that $m=1,(v_j=v_i)$. Set $G(v_i,1,r,1:v_i,-)=\infty$, and $G(v_i,0,r,1:v_i,-)=0$.
2. Let v_i be a non-leaf node, and let v_{i1} and v_{i2} be its two children. Suppose that $m=0$. Let $x=w_id(v_i,v_k)$. If $a(x,r)=1$ set $G(v_i,0,r,0:-,v_k)=\infty$. Otherwise, set

$$G(v_i,0,r,0:-,v_k)=G(v_{i1},0,r,0:-,v_k)+G(v_{i2},0,r,0:-,v_k).$$

For $q\geqslant 1$, set

$$G(v_i,q,r,0:-,v_k)=a(x,r)x+\min_{\substack{q_1+q_2=q-a(x,r)\\ |q_1|\leqslant|V_{i1}|\\ |q_2|\leqslant|V_{i2}|}}\{G(v_{i1},q_1,r,0:-,v_k)+G(v_{i2},q_2,r,0:-,v_k)\}.$$

Suppose that $m \geqslant 1$.

3. If $v_j = v_i$, set

$$G(v_i, q, r, m : v_i, -) = B,$$

where

$$B = \min_{\substack{q_1+q_2=q \\ m_1+m_2=m-1 \\ |q_1|,|m_1| \leqslant |V_{i1}| \\ |q_2|,|m_2| \leqslant |V_{i2}|}} \left\{ \min_{v_t \in V_{i1}} G(v_{i1}, q_1, r, m_1 : v_t, v_i) + \min_{v_s \in V_{i2}} G(v_{i2}, q_2, r, m_2 : v_s, v_i) \right\}.$$

4. If $v_j \neq v_i$, suppose without loss of generality that $v_j \in V_{i2}$.
5. Suppose that $d(v_i, v_j) < d(v_i, v_k)$. Define $x = w_i d(v_i, v_j)$. If $a(x, r) = 1$, and $q = 0$, set $G(v_i, 0, r, m : v_j, v_k) = \infty$. Otherwise, set

$$G(v_i, q, r, m : v_j, v_k) = a(x, r)x + B,$$

where

$$B = \min_{\substack{q_1+q_2=q-a(x,r) \\ m_1+m_2=m \\ |q_1|,|m_1| \leqslant |V_{i1}| \\ |q_2|,|m_2| \leqslant |V_{i2}| \\ |m_2| \geqslant 1}} \left\{ \min_{\substack{v_t \in V_{i1} \\ d(v_t,v_i) > d(v_i,v_j)}} G(v_{i1}, q_1, r, m_1 : v_t, v_j) + G(v_{i2}, q_2, r, m_2 : v_j, v_k) \right\}.$$

6. Suppose that $d(v_i, v_k) < d(v_i, v_j)$. Define $x = w_i d(v_i, v_k)$. If $a(x, r) = 1$ and $q = 0$, set $G(v_i, 0, r, m : v_j, v_k) = \infty$. Otherwise, set

$$G(v_i, q, r, m : v_j, v_k) = a(x, r)x + B,$$

where

$$B = \min_{\substack{q_1+q_2=q-a(x,r) \\ m_1+m_2=m \\ |q_1|,|m_1| \leqslant |V_{i1}| \\ |q_2|,|m_2| \leqslant |V_{i2}| \\ |m_2| \geqslant 1}} \left\{ \min_{\substack{v_t \in V_{i1} \\ d(v_t,v_i) > d(v_i,v_j)}} G(v_{i1}, q_1, r, m_1 : v_t, v_k) + G(v_{i2}, q_2, r, m_2 : v_j, v_k) \right\}.$$

To compute the complexity of the above scheme, note that for a fixed value of r, the total time needed to (recursively) evaluate $G(v_1, k, r, p : v_j, -)$ for all nodes v_j in $V = V_1$, (v_1 is assumed to be the root of the tree), is $O(k^2 p^2 n^3)$. (The optimal solution value to the respective p-facility k-centrum problem with service distance r is then given by $\min\{G(v_1, k, r, p : v_j, -) : v_j \in V\}$.)

A more careful analysis of the above scheme, using the results in Section 3 of [24], reveals that the complexity of computing $\min\{G(v_1, k, r, p : v_j, -) : v_j \in V\}$ is $O(kp^2 n^3)$, and $O(k^2 pn^3)$. Thus, the complexity bound is only $O((\min(k, p))kpn^3)$.

There are only $O(n^2)$ relevant values of the parameter r. These are the elements of the set $\{w_i d(v_i, v_j)\colon i, j = 1, \ldots, n\}$. Therefore, the discrete p-facility k-centrum model on a tree can be solved in $O((\min(k, p))kpn^5)$ time.

We note that the above algorithm can easily be modified to solve the generalized discrete model where there are setup costs for the facilities and the objective is to minimize the sum of the setup costs of the p facilities and the k largest service distances. The complexity will be the same.

The above complexity is significantly higher than the results known for some of the special cases. For the p-median problem ($k = n$), $O(p^2n^2)$ and $O(pn^2)$ algorithms are given in [18] and [24], respectively. For the p-center problem ($k = 1$), an $O(n \log^2 n)$ algorithm is presented in [20,9]. The general dynamic programming scheme (the bottom-up approach), that we have adopted above is similar to the approach used in [24] to solve the p-median problem. However, the k-centrum multi-facility problem is more complicated, and some of the ideas and properties utilized in [24] to derive the low complexity, do not seem to be applicable here. First, we had to introduce two additional parameters, q and r, in order to control the value of the qth largest distance, and be able to decompose the problem into the respective subproblems. (For the same reason, unlike the p-median model, where for the ith subproblem we could maintain only the closest facility to v_i, here we had to maintain the closest facility to v_i from "below" and "above".) Moreover, the recursive functions G, defined above, do not exhibit desirable monotonicity or convexity properties in q or r. Therefore, we had to solve $O(n^2)$ independent problems corresponding to the $O(n^2)$ relevant values of r. (See [14,15,25] for the use of such properties.)

3.3. *Solving the (continuous) multi-facility k-centrum problems on paths and trees*

It is shown in [18] that there is an optimal solution to the multi-facility center problem on tree graphs, where each facility is located either at a node or at a point which is at equal weighted distances from a pair of nodes. It can easily be shown that this result extends to the multi-facility k-centrum problem. (Note, however, that unlike the center problem, in the weighted case, the location point of a k-centrum which is at equal weighted distances from a pair of nodes, is not necessarily on the simple path connecting this pair of nodes.)

Consider first the continuous multi-facility k-centrum problem on a path. In this case, for each pair of nodes with positive weights, there are at most 2 points on the path that are at equal weighted distances from the two nodes. Therefore, we can discretize the problem and look at $O(n^2)$ potential sites for the facilities.

A similar discretization argument applies to the tree case. For each pair of nodes $\{v_i, v_j\}$, with $w_i \neq w_j$, and each edge of a tree, there is at most one point on the edge which is at equal weighted distances from the two nodes. (If $w_i = w_j$, the pair $\{v_i, v_j\}$ contributes only the middle point of the path connecting them as a potential site for a k-centrum.) Therefore, there are $O(n^3)$ potential sites for the facilities. In the unweighted case this number is only $O(n^2)$.

3.4. Solving degenerate models

We have assumed above that the problems that we solve satisfy a nondegeneracy assumption on the set of distances. This assumption can be removed by using a standard perturbation on the edge distances. Suppose that the edge set E is defined by $E = \{e_1, \ldots, e_m\}$. If ε denotes a small positive infinitesimal, we add the term ε^i to the length of edge e_i, $i = 1, \ldots, m$. For all values of ε, which are sufficiently small, the perturbed problem satisfies the nondegeneracy assumption. Therefore, we can apply the above algorithms symbolically to the perturbed problem, and obtain an optimal solution to the unperturbed problem, defined by $\varepsilon = 0$.

References

[1] P.K. Agarwal, B. Aronov, T.M. Chan, M. Sharir, On levels in arrangements of lines, segments, planes and triangles, Discrete Comput. Geom. 19 (1998) 315–331.
[2] G. Andreatta, F.M. Mason, k-eccentricity and absolute k-centrum of a tree, European J. Oper. Res. 19 (1985) 114–117.
[3] G. Andreatta, F.M. Mason, Properties of the k-centrum in a network, Networks 15 (1985) 21–25.
[4] Y. Bartal, On approximating arbitrary metrics by tree metrics, Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 161–168.
[5] M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection, J. Comput. Systems Sci. 7 (1973) 448–461.
[6] M. Charikar, C. Chekuri, A. Goel, S. Guha, Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median, Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 114–123.
[7] M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k-median problems, Proceedings of the 40th Annual Symposium on Foundations of Computer Science, 1999, pp. 378–388.
[8] M. Charikar, S. Guha, E. Tardos, D.B. Shmoys, A constant-factor approximation algorithm for the k-median problem, Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 1–10.
[9] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, J. ACM 34 (1987) 200–208.
[10] T.K. Dey, Improved bounds for planar k-sets and related problems, Discrete Comput. Geom. 19 (1998) 373–382.
[11] H. Edelsbrunner, E. Welzl, Constructing belts in two-dimensional arrangements with applications, SIAM J. Comput. 15 (1986) 271–284.
[12] D. Eppstein, Geometric lower bounds for parametric matroid optimization, Discrete Comput. Geom. 20 (1998) 463–476.
[13] M. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in network optimization algorithms, J. ACM 34 (1987) 596–615.
[14] R. Hassin, A. Tamir, Improved complexity bounds for location problems on the real line, Oper. Res. Lett. 10 (1991) 395–402.
[15] V.N. Hsu, T.J. Lowe, A. Tamir, Structured p-facility location problems on the line solvable in polynomial time, Oper. Res. Lett. 21 (1997) 159–164.
[16] K. Jain, V.V. Vazirani, Primal-dual approximation algorithms for metric facility location and k-median problems, Proceedings of the 40th Annual Symposium on Foundations of Computer Science 1999, pp. 1–10.
[17] O. Kariv, L.S. Hakimi, An algorithmic approach to network location problems. Part I: The p-centers, SIAM J. Appl. Math. 37 (1979) 513–538.
[18] O. Kariv, L.S. Hakimi, An algorithmic approach to network location problems. Part II: The p-medians, SIAM J. Appl. Math. 37 (1979) 539–560.

[19] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, J. ACM 30 (1983) 852–865.
[20] N. Megiddo, A. Tamir, New results on the complexity of p-center problems, SIAM J. Comput. 12 (1983) 751–758.
[21] P.H. Peeters, Some new algorithms for location problems on networks, European J. Oper. Res. 104 (1998) 299–309.
[22] M. Sharir, P.K. Agarwal, Davenport-Schinzel Sequences and Their Geometric Applications, Cambridge University Press, New York, 1995.
[23] P.J. Slater, Centers to centroids in a graph, J. Graph Theory 2 (1978) 209–222.
[24] A. Tamir, An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs, Oper. Res. Lett. 19 (1996) 59–64.
[25] R. Wilber, The concave least weight subsequence revisited, J. Algorithms 9 (1988) 418–425.