On a Tree-Shaped Facility Location Problem of Minieka

Ramon Rabinovitch
Department of Finance, College of Business Administration
University of Houston, Houston, Texas 77204

Arie Tamir
Department of Statistics and Operations Research
Stern School of Business
New York University, New York, New York 10003 and
Faculty of Exact Sciences, School of Mathematical Sciences
Tel Aviv University, Ramat-Aviv, Tel Aviv 69978, Israel

Minieka and Hakimi et al. considered the problem of locating a tree-shaped facility of a given length in a tree network, with the objective of maximizing the sum of node distances from this facility. They stated the complexity of the model as an open problem. We prove that it is NP-complete and provide a pseudopolynomial time algorithm.

Let $T = (V, E)$ be an undirected tree with node set $V = \{v_1, \ldots, v_n\}$ and edge set E. Each edge has a positive length and is assumed to be rectifiable. We refer to interior points on an edge by their distances (along the edge) from the two nodes of the edge. Let $A(T)$ denote the continuum set of points on the edges of T. The edge lengths induce a distance function on $A(T)$; for any x, y in $A(T)$, $d(x, y)$ will denote the length of the unique path connecting x and y. Also, for any subset $Y \subseteq A(T)$, $d(x, Y) = \text{Infimum}\ \{d(x, y) \mid y \in Y\}$. $A(T)$ is a metric space with respect to the above distance function.

A subset $Y \subseteq A(T)$ is a subtree of T if Y is both connected and closed. Since the edges are assumed to be rectifiable, we view a subtree Y as a finite (connected) collection of partial edges. Each partial edge corresponds to a subinterval of the interval representing that edge. We define $L(Y)$, the length of Y, to be the sum of the lengths of its partial edges. A subtree is said to be discrete if all its (relative) boundary points with respect to the distance function are nodes of T. It is almost discrete if at most one of its boundary points is not a node.

The following (obnoxious) location problem has been considered by Minieka [4] and Hakimi et al. [2]:

CCC 0028-3045/92/060515-08

Let B be a positive real number:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n} d(v_i, Y) \\ \text{s.t.} \quad & L(Y) = B \\ & Y \text{ is a subtree of } T. \end{aligned} \tag{1}$$

The complexity of this location model is stated as an open problem in [2, 4]. Minieka [4] has proved the following property which implies that the recognition version of (1) is in NP.

Lemma 1. *There exists an optimal solution subtree to* (1) *which is almost discrete.*

We prove that the above location problem is NP-hard by reducing the (NP-complete) partition problem to (1).

The Partition Model. Given positive integers $a_1, \ldots, a_n$, does there exist $S \subseteq \{1, 2, \ldots, n)$ such that $\sum_{i \in S} a_i = A/2$, where $A = \sum_{i=1}^{n} a_i$? (without loss of generality $a_i \leq A/2$, $i = 1, \ldots, n$).

Consider the star tree $T = (V, E)$ in Figure 1. T has $2n + 1$ nodes: the central node, and two more nodes on each "spoke" i, $i = 1, \ldots, n$. The length of each of the two edges on spoke i, $i = 1, \ldots, n$ is $a_i/2$.

Proposition 2. *There exists a solution set $S \subseteq \{1, \ldots, n\}$ to the partition problem, if and only if there exists a subtree Y of T in Figure 1, with*

$$L(Y) \geq \frac{A}{2}, \sum_{i=1}^{n} d(v_i, Y) \geq \frac{3}{4} A.$$

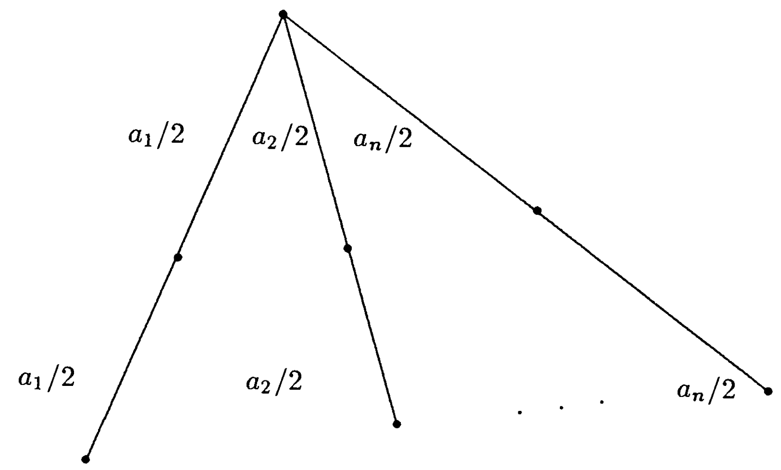

FIG. 1. The star tree.

Proof. Suppose first that $\sum_{i \in S} a_i = A/2$ for some $S \subseteq \{1, \ldots, n\}$. Consider the subtree Y consisting of all spokes $i \in S$. Then, $L(Y) = A/2$ and

$$\sum_{i=1}^{n} d(v_i, Y) = \frac{3}{2} \sum_{i \notin S} a_i = \frac{3}{4} A.$$

Next, suppose that Y is some subtree of length $A/2$ with node distance sum greater than or equal to $\frac{3}{4}A$. Since $a_i \leq A/2$, $i = 1, \ldots, n$, Y contains the central node of T in Figure 1. For $i = 1, \ldots, n$, let x_i denote the length of the subpath of Y that intersects spoke i, and let $f_i(x_i)$ denote the sum of the distances of the two nodes on spoke i from Y. For $i = 1, \ldots, n$, $f_i(x_i)$ is a convex function of x_i, as illustrated in Figure 2.

Using Lemma 1, we assume without loss of generality that there exists an index k such that $x_i \in \{0, a\}$ for all $i \neq k$. If $x_k \in \{0, a_k\}$, define $S = \{i \mid x_i = a_i\}$ to note that $\sum_{i \in S} a_i = A/2$. Thus, suppose that $x_k = ta_k$, where $0 < t < 1$. We will obtain a contradiction.

Define $S = \{i \mid x_i = a_i\}$. Then,

$$\sum_{i \in S} a_i + ta_k = \frac{A}{2} \tag{2}$$

$$\frac{3}{2} \sum_{\substack{i \notin S \\ i \neq k}} a_i + f_k(ta_k) \geq \frac{3}{4} A. \tag{3}$$

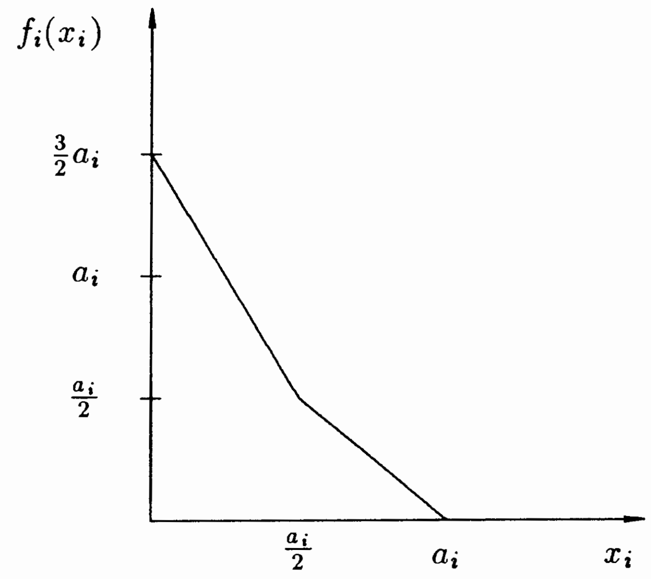

FIG. 2. The function $f_i(x_i)$.

Combining the convexity property, $f_k(ta_k) < \frac{3}{2}(1 - t)a_k$, with (3), we have

$$\frac{3}{2}\left(A - \sum_{i \in S} a_i - a_k\right) + \frac{3}{2}(1 - t)a_k > \frac{3}{4}A,$$

which is equivalent to

$$\frac{A}{2} - \sum_{i \in S} a_i - ta_k > 0. \tag{4}$$

However, (4) contradicts (2) and the proof is complete. ■

We note that if the maximization in (1) is replaced by a minimization, the problem can be solved in polynomial time as exhibited in [4]. However, if the selected subtree is required to be discrete, the minimization and the maximization versions of (1) are *NP*-hard, [2].

Having shown that the location model (1) is *NP*-hard, we next present a pseudopolynomial time algorithm solving the model in $O(B^2 n \log n)$ time. We assume that the edge lengths and the real parameter B in (1) are all positive integers.

To solve (1), we note from Lemma 1 that there is an optimal subtree containing a node of T. Thus, it suffices to solve a set of n subproblems where in each subproblem the subtree to be selected is restricted to contain some specified node. We will apply a more sophisticated decomposition approach that yields a better complexity bound. It is more convenient for this approach to consider the following weighted version of (1). Suppose that each node v_j, $j = 1, \ldots, n$, is associated with a positive weight w_j. Given a positive real B,

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n} w_i d(v_i, Y) \\ \text{s.t.} \quad & L(Y) = B \\ & Y \text{ is a subtree of } T. \end{aligned} \tag{5}$$

For each $j = 1, \ldots, n$, the restricted subproblem corresponding to node v_j is

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{n} w_i d(v_i, Y) \\ \text{s.t.} \quad & L(Y) = B \\ & Y \text{ is a subtree of } T \\ & \text{containing } v_j. \end{aligned} \tag{6}$$

As mentioned above, an optimal solution to (5) can be obtained by solving the n subproblems defined by (6). However, with the decomposition approach, we solve only one such subproblem on the original n node tree $T = (V, E)$ and then recursively solve (5) on two subtrees of T, say $T_1 = (V_2, E_1)$ and $T_2 = (V_2, E_2)$, which have only one node in common, and $|V_1| \leq \frac{2}{3}|V|$, $|V_2| \leq \frac{2}{3}|V|$. Thus, we decompose a problem on an n node tree into two subproblems on a tree with at most $\frac{2}{3}n$ nodes.

We focus on (6) and assume without loss of generality that $j = 1$. To facilitate the discussion, we root the tree T at node v_1. For each node $v \in V$, let $D(v)$ be the set of all nodes u having v on the unique path connecting them to the root v_1. $D(v)$ is the set of descendants of v. We also define $S(v)$, the set of sons of v, to be the set of descendants of v that are connected to v with an edge. [Note that v is in $D(v)$ but not in $S(v)$.] If $S(v)$ is empty, v is called a leaf of the rooted tree T.

To solve (6) on T, we recursively solve a sequence of subproblems defined on certain subtrees of T, starting with the leaves of T. To define these subtrees, consider a node v and suppose that $S(v) = \{v(1), v(2), \ldots, v(s(v))\}$, where $s(v) = |S(v)|$. For any $t = 1, \ldots, s(v)$, let $T(v, t)$ denote the subtree induced by the nodes in $\{v\} \cup N(v, t)$, where $N(v, t) = D(v(1)) \cup \cdots \cup D(v(t))$ (see Fig. 3).

For each nonleaf node v, $t = 1, \ldots, s(v)$, and integer x, define

$$\begin{aligned} f(v, t, x) = \text{Maximum} \quad & \sum_{v_j \in N(v,t)} w_j d(v_j, Y) \\ \text{s.t.} \quad & L(Y) = x \\ & Y \text{ is an almost discrete subtree of } T(v, t) \\ & \text{containing } v. \end{aligned}$$

The solution to (6) is then given by $f(v_1, s(v_1), B)$.

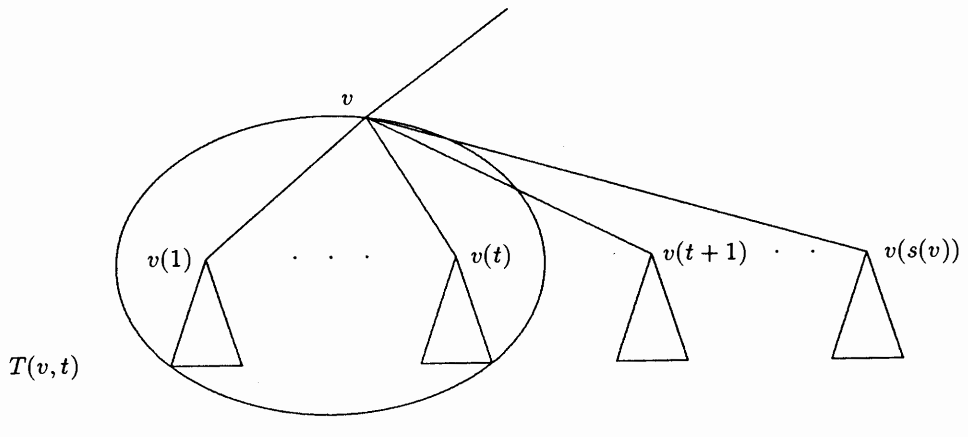

FIG. 3. The subtree $T(v, t)$.

To introduce the recursive equations for solving (6), we need another auxiliary function. For each node v, $t = 1, \ldots, s(v)$, and integer x, let $T^-(v, t)$ be the subtree induced by the nodes $\{v\} \cup D(v(t))$, and let

$$\begin{aligned} g(v, t, x) = \text{Maximum} \quad & \sum_{v_j \in D(v(t))} w_j d(v_j, Y) \\ \text{s.t.} \quad & L(Y) = x \\ & Y \text{ is an almost discrete subtree of } T^-(v, t) \\ & \text{containing } v. \end{aligned} \tag{7}$$

We now define the recursive equations. First, for any nonleaf node v and $1 \le t \le s(v)$,

$$g(v, t, a) = \begin{cases} \sum_{v_j \in D(v(t))} w_j(d(v_j, v) - a) & \text{if } a \le d(v, v(t)) \\ f(v(t), s(v(t)), a - d(v, v(t))) & \text{if } a > d(v, v(t)). \end{cases} \tag{8}$$

[a is an integer with $0 \le a \le L(T^-(v, t))$.] Next consider the recursive equations for $f(v, t, a)$. If $t = 1$,

$$f(v, 1, a) = g(v, 1, a). \tag{9}$$

Let $t > 1$. Then, for every integer a, $0 \le a \le \text{Min}(B, L(T(v, t)))$,

$$f(v, t, a) = \underset{\substack{0 \le b \le a \\ b \le L(T(v,t-1)) \\ a-b \le L(T^-(v,t)) \\ b \text{ integer}}}{\text{Maximum}} \{f(v, t-1, b) + g(v, t, a-b)\}. \tag{10}$$

[We note that Lemma 1 provides the justification for restricting the variable b in (10) to integer values.] To compute the complexity of the above procedure to solve (6), it will suffice to focus on (10). The effort to evaluate $f(v, t, a)$ for a specified triplet (v, t, a) is $O(a)$. Thus, to compute $f(v, t, a)$ for all $0 \le a \le \text{Min}(B, L(T(v, t)))$ for every node v and every integer t, $1 \le t \le s(v)$, the total effort is

$$O\left(B^2 \sum_{v \in V} s(v)\right) = O(B^2 n).$$

If we solve (6) for each node v_j, we obtain the solution to (5) in $O(B^2 n^2)$ time. Next we introduce a decomposition scheme that will reduce this complexity bound to $O(B^2 n \log n)$:

We first find a centroid node of the given tree, say node v_j [3]. The centroid decomposes the given tree $T = (V, E)$ into two subtrees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \{v_j\}$ and $|V_k| \le \frac{2}{3}|V|$, $k = 1, 2$. The effort to find a centroid of a tree is linear in $|V|$.

After we locate a centroid v_j of T, we solve the subproblem (6) corresponding to v_j in $O(B^2n)$ time using the above dynamic programming scheme. The solution obtained is a best solution to the original problem (5) among all subtrees containing v_j. Thus, if an optimal subtree does not contain v_j, it must be contained either in T_1 or in T_2. Therefore, it is sufficient to solve recursively the following two subproblems:

1. Consider the subtree $T_1 = (V_1, E_1)$. Redefine the weight w_j of the centroid v_j by

$$w_j = \sum_{v_k \in V_2} w_k$$

and

$$\begin{aligned} &\text{Maximize} && \sum_{v_i \in V_1} w_i d(v_i, Y) \\ &\text{s.t.} && L(Y) = B \\ & && Y \text{ is a subtree of } T_1. \end{aligned}$$

2. Consider the subtree $T_2 = (V_2, E_2)$. Redefine the weight w_j of the centroid v_j by

$$w_j = \sum_{v_k \in V_1} w_k$$

and

$$\begin{aligned} &\text{Maximize} && \sum_{v_i \in V_2} w_i d(v_i, Y) \\ &\text{s.t.} && L(Y) = B \\ & && Y \text{ is a subtree of } T_2. \end{aligned}$$

We now compute the total effort required to solve (5) with the recursive decomposition scheme. Let $q(n)$ denote the time needed to solve (5) on a tree with n nodes. Since we solve one restricted subproblem for the centroid of the n

node tree and two problems on trees with at most $\frac{2}{3}n$ nodes, we obtain the following inequality:

$$q(n) \leq cB^2n + q(n_1) + q(n_2), \tag{11}$$

where c is a constant independent of n and B, $n_1 \leq \frac{2}{3}n$, $n_2 \leq \frac{2}{3}n$, and $n_1 + n_2 = n + 1$. It is well known [1] that a solution to (11) satisfies $q(n) = O(B^2n \log n)$.

To conclude, we have shown that the problem of locating a tree-shaped facility of length B in an n node tree network, with the objective of maximizing the sum of node distances from this facility, is *NP*-hard. However, there exists an $O(B^2n \log n)$ dynamic programming decomposition algorithm for this problem. We also note that the latter algorithm can be used to develop a fully polynomial ε-approximation scheme for the problem. The technical details will be discussed elsewhere.

REFERENCES

[1] A. V. Aho, J. E. S. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley (1974).
[2] S. L. Hakimi, E. F. Schmeichel, and M. Labbe, On locating path or tree-shaped facilities on networks. *Networks,* to appear.
[3] N. Megiddo, A. Tamir, E. Zemel, and R. Chandrasekaran, An $O(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems. *SIAM J. Comput.* **10** (1981) 328–337.
[4] E. Minieka, The optimal location of a path or tree in a tree network. *Networks* **16** (1985) 309–321.

Received September 1990
Accepted January 1992