Improved Algorithms for the Minmax Regret 1-Median Problem

Hung-I Yu, Tzu-Chin Lin, and Biing-Feng Wang

Department of Computer Science, National Tsing Hua University Hsinchu, Taiwan 30043, Republic of China
herbert@cs.nthu.edu.tw, rems@cs.nthu.edu.tw, bfwang@cs.nthu.edu.tw

Abstract. This paper studies the problem of finding the 1-median on a graph where vertex weights are uncertain and the uncertainty is characterized by given intervals. It is required to find a minmax regret solution, which minimizes the worst-case loss in the objective function. Averbakh and Berman had an $O(mn^2 \log n)$-time algorithm for the problem on a general graph, and had an $O(n \log^2 n)$-time algorithm on a tree. In this paper, we improve these two bounds to $O(mn^2 + n^3 \log n)$ and $O(n \log n)$, respectively.

Keywords: Location theory, minmax regret optimization, medians.

1 Introduction

Over three decades, location problems on networks have received much attention from researchers in the fields of transportation and communication [9,10,11,12,17]. Traditionally, network location theory has been concerned with networks in which the vertex weights and edge lengths are known precisely. However, in practice, it is often impossible to make an accurate estimate of all these parameters [13,14]. Real-life data often involve a significant portion of uncertainty, and these parameters may change with time. Thus, location models involving uncertainty have attracted increasing research efforts in recent years [1,2,3,4,5,6,7,8,13,14,16,18,19,20,21].

Several ways for modeling network uncertainty have been defined and studied [13,16,18]. One of the most important models is the minmax regret approach, introduced by Kouvelis [13]. In the model, uncertainty of network parameters is characterized by given intervals, and it is required to minimize the worst-case loss in the objective function that may occur because of the uncertain parameters. During the last ten years, many important location problems have been studied on the minmax regret model. The 1-center problem was studied in [2,3,6], the p-center problem was studied in [2], and the 1-median problem was studied in [4,5,7,13].

The minmax regret 1-median problem is the focus of this paper. For a general graph with uncertain edge lengths, the problem is strongly NP-hard [1]. For a general graph with uncertain vertex weights, Averbakh and Berman [4] gave an $O(mn^2 \log n)$-time algorithm, where n is the number of vertices and m is the number of edges. As to trees, it was proved in [7] that uncertainty in edge lengths

D.Z. Chen and D.T. Lee (Eds.): COCOON 2006, LNCS 4112, pp. 52–62, 2006.
© Springer-Verlag Berlin Heidelberg 2006

can be ignored by setting the length of each edge to its upper bound. For a tree with uncertain vertex weights, Kouvelis et al. [13] proposed an $O(n^4)$-time algorithm. Chen and Lin [7] improved the bound to $O(n^3)$. Averbakh and Berman presented an $O(n^2)$-time algorithm in [4] and then improved it to $O(n\log^2 n)$ in [5]. In this paper, improved algorithms are presented for the minmax regret 1-median problem on a general graph and a tree with uncertain vertex weights. For general graphs, we improve the bound from $O(mn^2\log\, n)$ to $O(mn^2 + n^3\log\, n)$. For trees, we improve the bound from $O(n\log^2 n)$ to $O(n\log\, n)$.

The remainder of this paper is organized as follows. In Section 2, notation and definitions are introduced. In Sections 3 and 4, improved algorithms for the minmax regret 1-median problem on a general graph and a tree are proposed, respectively. Finally, in Section 5, we conclude this paper.

2 Notation and Definitions

Let $G = (V, E)$ be an undirected connected graph, where V is the vertex set and E is the edge set. Let $n = |V|$ and $m = |E|$. In this paper, G also denotes the set of all points of the graph. Thus, the notation $x \in G$ means that x is a point along any edge of G which may or may not be a vertex of G. Each edge $e \in E$ has a nonnegative length. For any two points $a, b \in G$, let $d(a, b)$ be the distance of the shortest path between a and b. Suppose that the matrix of shortest distances between vertices of G is given. Each vertex $v \in V$ is associated with two positive values w_v^- and w_v^+, where $w_v^- \leq w_v^+$. The weight of each vertex $v \in V$ can take any value randomly from the interval $[w_v^-, w_v^+]$. Let Σ be the Cartesian product of intervals $[w_v^-, w_v^+], v \in V$. Any element $S \in \Sigma$ is called a scenario and represents a feasible assignment of weights to the vertices of G. For any scenario $S \in \Sigma$ and any vertex $v \in V$, let w_v^S be the weight of v under the scenario S.

For any scenario $S \in \Sigma$, and a point $x \in G$, we define

$$F(S, x) = \sum_{v \in V} (w_v^S \times d(v, x)),$$

which is the total weighted distance from all the vertices to x according to S. Given a specific scenario $S \in \Sigma$, the classical 1-median problem is to find a point $x^* \in G$ that minimizes $F(S, x^*)$. The point x^* is called a 1-median of G under the scenario S. For any point $x \in G$, the regret of x with respect to a scenario $S \in \Sigma$ is $max_{y \in G} F(S, x) - F(S, y)$ and the maximum regret of x is

$$Z(x) = max_{S \in \Sigma} max_{y \in G} F(S, x) - F(S, y).$$

The minmax regret 1-median problem is to find a point $x \in G$ minimizing $Z(x)$.

3 Minmax Regret 1-Median on a General Graph

Averbakh and Berman [4] had an $O(mn^2\log\, n)$-time algorithm to find a minmax regret 1-median of a graph $G = (V, E)$. In this section, we give a new implementation of their algorithm, which requires $O(mn^2 + n^3\log\, n)$ time.

3.1 Averbakh and Berman's Algorithm

It is well-known that there is always a vertex that is a solution to the classical 1-median problem [10]. Thus, for any scenario $S \in \Sigma$, $min_{y \in G} F(S, y) = min_{y \in V} F(S, y)$. Therefore, the regret of x with respect to a scenario $S \in \Sigma$ can also be expressed as $max_{y \in V} F(S, x) - F(S, y)$. Consequently, we have

$$\begin{aligned} Z(x) &= max_{S \in \Sigma} max_{y \in V} F(S, x) - F(S, y) \\ &= max_{y \in V} max_{S \in \Sigma} F(S, x) - F(S, y). \end{aligned}$$

For any point $x \in G$ and vertex $y \in V$, define $R(x, y) = max_{S \in \Sigma} F(S, x) - F(S, y)$. Then, we have

$$Z(x) = max_{y \in V} R(x, y).$$

The problem is to find a point $x \in G$ that minimizes $Z(x)$. For ease of presentation, only the computation of $min_{x \in G} Z(x)$ is described. For each edge $e \in E$, let $Z_e^* = min_{x \in e} Z(x)$. Averbakh and Berman solved the minmax regret 1-median problem by firstly determining Z_e^* for every $e \in E$ and then computing $min_{x \in G} Z(x)$ as the minimum among all Z_e^*. Their computation of each Z_e^* takes $O(n^2 \log n)$ time and thus their solution to the minmax regret 1-median problem requires $O(mn^2 \log n)$ time. Let $e = (a, b)$ be an edge in G. In the remainder of this section, Averbakh and Berman's algorithm for computing Z_e^* is described. For ease of description, e is regarded as an interval $[a, b]$ on the real line so that any point on e corresponds to a real number $x \in [a, b]$.

Consider the function $R(\cdot, y)$ for a fixed $y \in V$. For a point $x \in e$, any scenario $S \in \Sigma$ that maximizes $F(S, x) - F(S, y)$ is called a worst-case scenario of x according to y. For each point $x \in e$ and each vertex $v \in V$, let

$$w_v^{x,y} = \begin{cases} w_v^+ & \text{if } d(v, x) > d(v, y), \text{ and} \\ w_v^- & \text{if } d(v, x) \leq d(v, y). \end{cases}$$

For each point $x \in e$, let $S_{(x,y)}$ be the scenario in which the weight of each $v \in V$ is $w_v^{x,y}$. It is easy to see that $S_{(x,y)}$ is a worst-case scenario of x according to y. Thus, we have the following lemma.

Lemma 3.1 [4]. $R(x, y) = F(S_{(x,y)}, x) - F(S_{(x,y)}, y)$.

For each $v \in V$, let x_v be the point on e that is farthest from v. For convenience, each x_v is called a *pseudo-node* of e. There are at most two points $x \in [a, b]$ with $d(v, x) = d(v, y)$. For ease of presentation, assume that there are two such points y_1^v and y_2^v, where $y_1^v < y_2^v$. For convenience, y_1^v and y_2^v are called *y-critical points* of e. In the interval $[a, b]$, $R(\cdot, y)$ is a piecewise linear function having $O(n)$ breakpoints, including at most n pseudo-nodes and at most $2n$ y-critical points. These breakpoints can be easily determined in $O(n)$ time. Let $B(y)$ be the non-decreasing sequence of the pseudo-nodes and y-critical points on e. Averbakh and Berman showed the following.

Lemma 3.2 [4]. *Let $y \in V$ be a vertex. Given $B(y)$, the function $R(\cdot, y)$ on e can be constructed in $O(n)$ time.*

Averbakh and Berman computed Z_e^* as follows. First, by sorting, $B(y)$ is obtained in $O(n^2 \log n)$ time for every $y \in V$. Next, the function $R(\cdot, y)$ on e is constructed for every $y \in V$, which takes $O(n^2)$ time. Finally, $Z_e^* = min_{x \in e} Z(x) = min_{x \in e}\{max_{y \in V} R(x, y)\}$ is computed in $O(n^2)$ time, by using Megiddo's linear-time algorithm for two-variable linear programming [17].

3.2 The Improved Algorithm

Averbakh and Berman's algorithm for computing each Z_e^* requires $O(n^2 \log n)$ time. The bottleneck is the computation of $B(y)$ for each $y \in V$. In this subsection, we show that with a simple $O(n^3 \log n)$-time preprocessing, each $B(y)$ can be computed in $O(n)$ time.

The preprocessing computes a list for each edge $e \in E$ and computes n lists for each vertex $a \in V$. The list computed for each $e \in E$ is $P(e)$, which is the non-decreasing sequence of all pseudo-nodes of e. The n lists computed for each $a \in V$ are, respectively, denoted by $Y(a, y)$, $y \in V$. Each $Y(a, y)$, $y \in V$, stores the non-decreasing sequence of the values $d(v, y) - d(v, a)$ of all $v \in V$. By using sorting and with the help of the distance matrix, the computation of each $P(e)$ and each $Y(a, y)$ takes $O(n \log n)$ time. Thus, the preprocessing requires $O((m + n^2)n \log n) = O(n^3 \log n)$ time.

Let $e = (a, b)$ be an edge. Consider the function $R(\cdot, y)$ for a fixed $y \in V$ on e. In the following, we show how to compute $B(y)$ in $O(n)$ time. The function $R(\cdot, y)$ has two kinds of breakpoints: pseudo-nodes and y-critical points. As mentioned, for each vertex $v \in V$ there are at most two y-critical points $x \in [a, b]$. To be more specific, if $d(v, a) < d(v, y) < d(v, x_v)$, there is a point x in the open interval (a, x_v) with $d(v, x) = d(v, y)$. In such a case, we say that v generates a *left y-critical point* and denote the point as y_1^v. On the other hand, if $d(v, b) < d(v, y) < d(v, x_v)$, we say that v generates a *right y-critical point* and denote the point as y_2^v. We obtain $B(y)$ as follows. First, we determine the y-critical points generated by all vertices $v \in V$. Then, we compute Y_1 as the non-decreasing sequence of the left y-critical points. Since the slope of $d(v, x)$ in the interval $[a, x_v]$ is $+1$, it is easy to conclude that each left y-critical point y_1^v is at a distance of $d(v, y) - d(v, a)$ from a. The sequence $Y(a, y)$ stores the non-decreasing sequence of the values $d(v, y) - d(v, a)$ of all $v \in V$. Thus, the order of the left y-critical points in Y_1 can be obtained from $Y(a, y)$ in $O(n)$ time by a simple scan. Next, we compute Y_2 as the non-decreasing sequence of the right y-critical points. Since the slope of $d(v, x)$ in the interval $[x_v, b]$ is -1, each right y-critical point y_2^v is at a distance of $d(v, y) - d(v, b)$ from b. The sequence $Y(b, y)$ stores the non-decreasing sequence of the values $d(v, y) - d(v, b)$ of all $v \in V$. Thus, the order of the right y-critical points in Y_2 can be obtained by scanning the reverse sequence of $Y(b, y)$ in $O(n)$ time. Finally, $B(y)$ is computed in $O(n)$ time by merging Y_1, Y_2, and $P(e)$.

With the above computation of $B(y)$, each Z_e^* can be computed in $O(n^2)$ time. Thus, we obtain the following.

Theorem 3.3. *The minmax regret 1-median problem on a general graph can be solved in $O(mn^2 + n^3 log\ n)$ time.*

We remark that Averbakh and Berman's algorithm uses $O(n^2)$ space, whereas ours uses $O(n^3)$ space.

4 Minmax Regret 1-Median on a Tree

In Subsection 4.1, Averbakh and Berman's $O(n\log^2 n)$-time algorithm for the minmax regret 1-median problem on a tree $T = (V, E)$ is described. In Subsection 4.2, an $O(n\log n)$-time improved algorithm is presented.

4.1 Averbakh and Berman's Algorithm

An edge that contains a minmax regret 1-median is called an optimal edge. Averbakh and Berman's algorithm consists of two stages. The first stage finds an optimal edge e^*, which requires $O(n\log^2 n)$ time. The second stage finds the exact position of a minmax regret 1-median on e^* in $O(n)$ time. Our improvement is obtained by giving a more efficient implementation of the first stage. Thus, only the first stage is described in this subsection.

Let $v \in V$ be a vertex. By removing v and its incident edges, T is broken into several subtrees, each of which is called an *open v-branch*. For each open v-branch X, the union of v, X, and the edge connecting v and X is called a *v-branch*. For any vertex $p \in v$ in T, let $B(v, p)$ be the v-branch containing p. Let S^- be the scenario in which the weight of every $v \in V$ is w_v^-. Let S^+ be the scenario in which the weight of every $v \in V$ is w_v^+. For any subtree X of T, let $V(X)$ be the set of vertices in X. For any vertex $a \in V$ and any open a-branch X, the following auxiliary values are defined:

$$W^+(X) = \Sigma_{v\in V(X)} w_v^+ \qquad W^-(X) = \Sigma_{v\in V(X)} w_v^-$$
$$D^+(X,a) = \Sigma_{v\in V(X)} w_v^+ \times d(v,a) \qquad D^-(X,a) = \Sigma_{v\in V(X)} w_v^- \times d(v,a)$$
$$F^+(a) = \Sigma_{v\in V} w_v^+ \times d(v,a) \qquad F^-(a) = \Sigma_{v\in V} w_v^- \times d(v,a)$$

By using dynamic programming, these auxiliary values of all vertices a and all open a-branches X are pre-computed, which takes $O(n)$ time [5].

Let $R(x, y)$ and $S_{(x,y)}$ be defined the same as in Subsection 3.1. Consider the computation of $R(x, y)$ for a pair of vertices x, $y \in V$. An edge is a bisector of a simple path if it contains the middle point of the path. In case the middle point is located at a vertex, both the edges connecting the vertex are bisectors. Let $(i, j) \in E$ be a bisector of the path from x to y, where i is closer to x than j. Let X be the open j-branch containing x and Y be the open i-branch containing y. By Lemma 3.1, $R(x, y) = F(S_{(x,y)}, x) - F(S_{(x,y)}, y)$. Since T is a tree, it is easy to see that under the scenario $S_{(x,y)}$ the weights of all vertices in Y are equal to their upper bounds w_v^+ and the weights of all vertices in X are equal to their lower bounds w_v^-. Thus, $F(S_{(x,y)}, x)$ and $F(S_{(x,y)}, y)$ can be computed in $O(1)$ time according to the following equations [5]:

$$F(S_{(x,y)}, x) = F^-(x) - D^-(Y, i) - d(i, x) \times W^-(Y) + D^+(Y, i) + d(i, x) \times W^+(Y)$$
$$F(S_{(x,y)}, y) = F^+(y) - D^+(X, j) - d(j, y) \times W^+(X) + D^-(X, j) + d(j, y) \times W^-(X)$$

Based upon the above discussion, the following lemma is obtained.

Lemma 4.1 [5]. *Let $x \in V$ be a vertex. Given the bisectors of the paths from x to all the other vertices, $R(x, y)$ can be computed in $O(n)$ time for all $y \in V$.*

For each $x \in V$, let $\hat{y}(x)$ be a vertex in T such that $R(x, \hat{y}(x)) = max_{y \in V} R(x, y)$. We have the following.

Lemma 4.2 [5]. *Let $x \in V$ be a vertex. If $x = \hat{y}(x)$, x is a minmax regret 1-median of T; otherwise, $B(x, \hat{y}(x))$ contains a minmax regret 1-median.*

A *centroid* of T is a vertex $c \in V$ such that every open c-branch has at most $n/2$ vertices. It is easy to find a centroid of T in $O(n)$ time [12]. Averbakh and Berman's algorithm for finding an optimal edge is as follows.

Algorithm 1. OPTIMAL_EDGE(T)
begin
1 $\hat{T} \leftarrow T$ // $\hat{T}$ is the range for searching an optimal edge
2 **while** ($\hat{T}$ is not a single edge) **do**
3 **begin**
4 $x \leftarrow$ a centroid of $\hat{T}$
5 **for** each $y \in V$ **do** compute a bisector of the path from x to y
6 **for** each $y \in V$ **do** $R(x, y) \leftarrow F(S_{(x,y)}, x) - F(S_{(x,y)}, y)$
7 $\hat{y}(x) \leftarrow$ the vertex that maximizes $R(x, y)$ over all $y \in V$
8 **if** $x = \hat{y}(x)$ **then return** (x) // x is a minmax regret 1-median
9 **else** $\hat{(T)} \leftarrow B(x, \hat{y}(x)) \cap \hat{T}$ // reduce the search range to a subtree
10 **end**
11 **return** $\hat{T}$ // $\hat{T}$ is an optimal edge
end

The while-loop in Lines 2-10 performs $O(\log\ n)$ iterations, in which Line 5 requires $O(n\log\ n)$ time [5] and is the bottleneck. Therefore, Algorithm 1 finds an optimal edge in $O(n\log^2 n)$ time.

4.2 An Improved Algorithm

For each $e = (i, j) \in E$, define S_e as the scenario in which the weight of every vertex v in the open i-branch containing j is w_v^+ and the weight of every vertex v in the open j-branch containing i is w_v^-. A key procedure of the new algorithm is to compute for every $e \in E$ a vertex that is a classical 1-median under the scenario S_e. The computation is described firstly in Subsection 4.2.1. Then, the new algorithm is proposed in Subsection 4.2.2.

4.2.1 Computing Medians

For ease of discussion, throughout this subsection, we assume that T is rooted at an arbitrary vertex $r \in V$. Let $v \in V$ be a vertex. Denote $P(v)$ as the path

from r to v, $p(v)$ as the parent of v, and T_v as the subtree of T rooted at v. For convenience, the subtrees rooted at the children of v are called *subtrees of* v. For any subtree X of T, define the *weight* of X under a scenario $S \in \Sigma$ as $W(S, X) = \sum_{v \in V(X)} w_v^S$. For any scenario $S \in \Sigma$ and $v \in V$, define

$$\delta(S, v) = 2W(S, T_v) - W(S, T).$$

Note that $2W(S, T_v) - W(S, T) = W(S, T_v) - W(S, T \backslash T_v)$ and thus $\delta(S, v)$ is equal to the weight difference between T_v and its complement under the scenario S. By the definition of $\delta(S, v)$, it is easy to see the following.

Lemma 4.3. *For any scenario $S \in \Sigma$, the function value of $\delta(S, \cdot)$ is decreasing along any downward path in T.*

Under a scenario $S \in \Sigma$, a vertex v is a classical 1-median of T if and only if $W(S, X) \leq 1/2 \times W(S, T)$ for all open v-branches X [12]. By using this property, the following lemma can be obtained.

Lemma 4.4 [15]. *For any scenario $S \in \Sigma$, the vertex $v \in V$ with the smallest $\delta(S, v) > 0$ is a classical 1-median.*

Under a scenario $S \in \Sigma$, T may have more than one median, but it has a unique vertex v with the smallest $\delta(S, v) > 0$, which is called *the median*, denoted as $m(S)$. For any $v \in V$, define the *heavy path* of v under a scenario $S \in \Sigma$, denoted by $h(S, v)$, as the path starting at v and at each time moving down to the heaviest subtree, with a tie broken arbitrarily, until a leaf is reached. With some efforts, the following can be proved.

Lemma 4.5. *Let $S \in \Sigma$ be a scenario and $v \in V$ be a vertex with $\delta(S, v) > 0$. Then, $h(S, v)$ contains the median $m(S)$.*

Lemma 4.3, 4.4 and 4.5 are useful for finding a classical 1-median for a fixed scenario $S \in \Sigma$. In our problem, all scenarios S_e, $e \in E$, need to be considered. Let $e = (i, j)$ be an edge in T such that i is the parent of j. For ease of presentation, denote $S_{\bar{e}}$ as the scenario $S_{(j,i)}$. For each $v \in V$, the weight of T_v and the heavy path of v only depend on the weight assignment to the vertices in T_v. Therefore, the following two lemmas can be obtained.

Lemma 4.6. *Let $e = (i, j)$ be an edge in T such that i is the parent of j. For any $v \in V$,*

$$W(S_e, T_v) = \begin{cases} W(S^+, T_v) & \text{if } v \in T_j, \\ W(S^-, T_v) & \text{if } v \notin T_j \text{ and } v \notin P(i), \\ W(S^-, T_v) - W(S^-, T_j) + W(S^+, T_j) & \text{if } v \in P(i); \text{ and} \end{cases}$$

$$W(S_{\bar{e}}, T_v) = \begin{cases} W(S^-, T_v) & \text{if } v \in T_j, \\ W(S^+, T_v) & \text{if } v \notin T_j \text{ and } v \notin P(i), \\ W(S^+, T_v) - W(S^+, T_j) + W(S^-, T_j) & \text{if } v \in P(i); \text{ and} \end{cases}$$

Lemma 4.7. *Let $e = (i, j)$ be an edge in T such that i is the parent of j. For any $v \notin P(i)$,*

$$h(S_e, v) = \begin{cases} h(S^+, v) & \text{if } v \in T_j, \\ h(S^-, v) & \text{if } v \notin T_j; \end{cases} \text{ and } h(S_{\bar{e}}, v) = \begin{cases} h(S^-, v) & \text{if } v \in T_j, \\ h(S^+, v) & \text{if } v \notin T_j. \end{cases}$$

According to Lemmas 4.6 and 4.7, maintaining information of T under the scenarios S^+ and S^- is very useful to the computation of $m(S_e)$ for every $e \in E$. With a bottom-up computation, we pre-compute the values $W(S^+, T_v)$ and $W(S^-, T_v)$ for all $v \in V$ in $O(n)$ time. With the pre-computed values, for any $e \in E$ and $v \in V$, $\delta(S_e, v) = 2W(S_e, T_v) - W(S_e, T_r)$ can be determined in $O(1)$ time by Lemmas 4.6. Besides, we preprocess T to construct the heavy paths $h(S^+, v)$ and $h(S^-, v)$ for every $v \in V$. The total length of the heavy paths of all $v \in V$ is $O(n^2)$. However, they can be constructed in $O(n)$ time and stored in $O(n)$ space [22].

Let $H^+ = \{h(S^+, v) | v \in V\}$ and $H^- = \{h(S^-, v) | v \in V\}$. Let $e \in E$ be an edge. Since the value of $\delta(S_e, \cdot)$ decreases along any downward path in T, we can compute the median $m(S_e)$ by firstly finding a path in $H^+ \cup H^-$ that contains the median and then locating the median by performing binary search on the path. The following two lemmas help in the finding. Due to the page limit, the proofs are omitted.

Lemma 4.8. *Let $e = (i, j)$ be an edge in T such that i is the parent of j. Let z be the lowest vertex on $P(j)$ with $\delta(S_e, z) > 0$. If $z = j$, $h(S^+, z)$ contains $m(S_e)$; otherwise, $h(S^-, z)$ contains $m(S_e)$.*

Lemma 4.9. *Let $e = (i, j)$ be an edge in T such that i is the parent of j. Let z be the lowest vertex on $P(j)$ with $\delta(S_{\bar{e}}, z) > 0$. If $z = j$, $h(S^-, z)$ contains $m(S_{\bar{e}})$; otherwise, either $h(S^+, z)$ or $h(S^+, c')$ contains $m(S_{\bar{e}})$, where c' is the child of z such that $T_{c'}$ is the second heaviest under the scenario S^+.*

Now, we are ready to present our algorithm for computing $m(S_e)$ for all $e \in E$. It is as follows.

Algorithm 2. COMPUTE_MEDIANS(T)
begin
1 Orient T into a rooted tree with an arbitrary root $r \in V$
2 Compute $W(S^+, T_v)$, $W(S^-, T_v)$, $h(S^+, v)$, and $h(S^-, v)$ for all $v \in V$
3 **for** each $v \in V$ **do**
4 $c'(v) \leftarrow$ the child of v such that $T_{c'(v)}$ is the second heaviest under S^+
5 **for** each $j \in V - \{r\}$ **do** (in depth-first search order)
6 **begin**
7 $e \leftarrow (p(j), j)$
8 $P(j) \leftarrow$ the path from r to j
9 $m(S_e) \leftarrow$ the classical 1-median under S_e
10 $m(S_{\bar{e}}) \leftarrow$ the classical 1-median under $S_{\bar{e}}$
11 **end**
12 **return** $(\{m(S_e) | e \in E\})$
end

The time complexity of Algorithm 2 is analyzed as follows. Lines 1-4 requires $O(n)$ time. Consider the for-loop in Lines 5-11 for a fixed $j \in V - r$. Line 7 takes

$O(1)$ time. Since the vertices are processed in depth-first search order, Line 8 also takes $O(1)$ time. Based upon Lemma 4.8, Line 9 is implemented in $O(\log n)$ time as follows. First, we compute z as the lowest vertex on $P(j)$ with $\delta(S_e, z) > 0$. Then, if $z = j$, we find $m(S_e)$ on $h(S^+, z)$; otherwise, we find $m(S_e)$ on $h(S^-, z)$. The computation of z and the finding of $m(S_e)$ are done in $O(\log n)$ time by using binary search. Similarly, based upon Lemma 4.9, Line 10 is implemented in $O(\log n)$ time. Therefore, each iteration of the for-loop takes $O(\log n)$ time. There are $n - 1$ iterations. Thus, the for-loop requires $O(n\log n)$ time in total. We obtain the following.

Theorem 4.10. *The computation of $m(S_e)$ of all $e \in E$ can be done in $O(n\log n)$ time.*

4.2.2 The New Approach

We pre-compute the medians $m(S_e)$ for all $e \in E$. Also, as in Subsection 4.1, the auxiliary values $W^+(X)$, $W^-(X)$, $D^+(X, a)$, $D^-(X, a)$, $F^+(a)$, and $F^-(a)$ of all vertices $a \in V$ and all open a-branches X are pre-computed.

Averbakh and Berman's algorithm finds an optimal edge by repeatedly reducing the search range $\hat{T}$ into a smaller subtree until only one edge is left. Let x be the centroid of the current search range $\hat{T}$. The reduction is done by determining a vertex $\hat{y}(x)$ with $R(x, \hat{y}(x)) = max_{y \in V} R(x, y)$ and then reducing $\hat{T}$ into $B(x, \hat{y}(x)) \cap \hat{T}$. A solution to the finding of bisectors is required for their determination of $\hat{y}(x)$. The new algorithm is obtained by using a different approach for the determination of $\hat{y}(x)$, which is based upon the following lemma. We omit the proof due to the page limit.

Lemma 4.11. *For any $x \in V$, there is an edge $e \in E$ such that $R(x, m(S_e)) = F(S_e, x) - F(S_e, m(S_e)) = max_{y \in V} R(x, y)$.*

According to Lemma 4.11, we modify Algorithm 1 by replacing Lines 5-7 with the following.

5 **for** each $e \in E$ **do** $R'(e) \leftarrow F(S_e, x) - F(S_e, m(S_e))$
6 $\hat{e} \leftarrow$ the edge that maximizes $R'(e)$ over all $e \in E$
7 $\hat{y}(x) \leftarrow m(S_{\hat{e}})$

Let $e = (i, j)$ be an edge in T. Let X be the open j-branch containing i and Y be the open i-branch containing j. Since T is a tree, for every $v \in V(X)$, we have $F(S_e, v) = F^-(v) - D^-(Y, i) - d(i, v) \times W^-(Y) + D^+(Y, i) + d(i, v) \times W^+(Y)$; and for every $v \in V(Y)$, we have $F(S_e, v) = F^+(v) - D^+(X, j) - d(j, v) \times W^+(X) + D^-(X, j) + d(j, v) \times W^-(X)$. By using these two equations, it is easy to implement the new Line 5 in $O(n)$ time. The new Lines 6 and 7 take $O(n)$ time. Therefore, Algorithm 2 requires $O(n\log n)$ time after the replacement.

Theorem 4.12. *The minmax regret 1-median problem on a tree can be solved in $O(n\log n)$ time.*

5 Concluding Remarks

During the last decade, minmax regret optimization problems have attracted significant research efforts. In [22], an $O(mn\log n)$-time algorithm is obtained for the minmax regret 1-center problem on a general graph, and an $O(n\log^2 n)$-time algorithm is obtained for the problem on a tree. For many location problems, however, there are still gaps between the time complexities of the solutions to their classical versions and those to their minmax regret versions. It would be a great challenge to bridge these gaps.

References

1. Averbakh, I.: On the complexity of a class of robust location problems. Working Paper. Western Washington University. Bellingham, WA. (1997)
2. Averbakh, I., Berman, O.: Minimax regret p-center location on a network with demand uncertainty. *Location Science* **5** (1997) 247–254
3. Averbakh, I., Berman, O.: Algorithms for the robust 1-center problem on a tree. *European Journal of Operational Research* **123** (2000) 292–302
4. Averbakh, I., Berman, O.: Minmax regret median location on a network under uncertainty. *Informs Journal on Computing* **12** (2000) 104–110
5. Averbakh, I., Berman, O.: An improved algorithm for the minmax regret median problem on a tree. *Networks* **41** (2003) 97–103
6. Burkard, R. E., Dollani, H.: A note on the robust 1-center problem on trees. *Annals of Operations Research* **110** (2002) 69–82
7. Chen, B. T., Lin, C. S.: Minmax-regret robust 1-median location on a tree. *Networks* **31** (1998) 93–103
8. Drezner, Z.: Sensitivity analysis of the optimal location of a facility. *Naval Research Logistics Quarterly* **33** (1980) 209–224
9. Goldman, A. J.: Optimal center location in simple networks. *Transportation Science* **5** (1971) 212–221
10. Hakimi, S. L.: Optimal locations of switching centers and the absolute centers and medians of a graph. *Operations Research* **12** (1964) 450–459
11. Kariv, O., Hakimi, S. L.: An algorithmic approach to network location problems. I: The p-centers. *SIAM Journal on Applied Mathematics* **37** (1979) 513–538
12. Kariv, O., Hakimi, S. L.: An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics* **37** (1979) 539–560
13. Kouvelis, P., Vairaktarakis, G., Yu, G.: Robust 1-median location on a tree in the presence of demand and transportation cost uncertainty. Working Paper 93/94-3-4. Department of Management Science and Information Systems, Graduate School of Business, The University of Texas at Austin (1994)
14. Kouvelis, P., Yu, G.: Robust discrete optimization and its applications. Kluwer Academic Publishers, Dordrecht (1997)
15. Ku, S. C., Lu, C. J., Wang, B. F., Lin, T. C.: Efficient algorithms for two generalized 2-median problems on trees. in *Proceedings of the 12th International Symposium on Algorithms and Computation* (2001) 768–778
16. Labbe, M., Thisse, J.-F., Wendell, R.: Sensitivity analysis in minisum facility location problems. *Operations Research* **38** (1991) 961–969
17. Megiddo, N.: Linear-time algorithms for linear-programming in R^3 and related problems. *SIAM Journal on Computing* **12** (1983) 759–776

18. Mirchandani, P. B., Odoni, A. R.: Location of medians on stochastic networks. *Transportation Science* **13** (1979) 85–97
19. Mirchandani, P. B., Oudjit, A., Wong, R. T.: Multidimensional extensions and a nested dual approach for the *M*-median problem. *European Journal of Operational Research* **21** (1985) 121–137
20. Oudjit, A.: Median locations on deterministic and probabilistic multidimensional networks. PhD Dissertation. Rennselaer Polytechnic Institute, Troy (1981)
21. Weaver, J. R., Church, R. L.: Computational procedures of location problems on stochastic networks. *Transportation Science* **17** (1983) 168–180
22. Yu, H. I, Lin, T. C., Wang, B. F.: Improved Algorithms for the Minmax Regret Single-Facilty Problems. Manuscript (2005)