On the Minmax Regret Path Center Problem on Trees

Biing-Feng Wang(✉), Jhih-Hong Ye, and Chih-Yu Li

Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan, Republic of China
{bfwang, jhong, cyuli}@cs.nthu.edu.tw

Abstract. This paper studies the problem of finding the path center on a tree in which vertex weights are uncertain and the uncertainty is described by given intervals. It is required to find a minmax regret solution, which minimizes the worst-case loss in the objective function. An $O(n \log n)$-time algorithm is presented, improving the previous upper bound of $O(n^2)$.

1 Introduction

The objective of a location problem is to decide the location of facilities in a network so as to minimize the communication or transportation costs [14, 15, 22, 24]. A network usually involves two types of parameters: weights of nodes and lengths of edges. Traditionally, the node weights and edge lengths of a network are assumed to be known precisely. In real transportation systems, the weights and lengths of a network may fluctuate or be inaccurate due to poor measurements. Thus, location models involving uncertainty have attracted significant research efforts [11, 12, 17, 23]. One of the most important ways for modeling network uncertainty is the *minmax regret approach*, introduced by Kouvelis and Yu [16]. 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.

Minmax regret location problems have received considerable attention in the past two decades. In network location theory, the shapes of facilities can be points, paths, or trees. Path- and tree-shaped facilities are called extensive facilities [18]. For point-shaped facility problems, most important ones have been studied comprehensively on the minmax regret model [3–6, 8, 9, 16, 31]. However, for extensive facility problems, there are only a few results on the minmax regret model, although there are considerable results on the classical model [7, 18, 20, 26–28]. In a breakthrough paper by Puerto et al. [21], polynomial algorithms were presented for the following three important path-shaped problems: the minmax regret path center, path median, and path centdian problems. Since these problems are NP-hard on general networks, their work was confined to trees. The time complexities of their algorithms are, respectively, $O(n^2)$, $O(n^4)$, and $O(n^5 \log n)$. In [29, 30] the upper bounds of the minmax regret path median and path centdian problems were improved to $O(n^2)$ and $O(n^4)$, respectively.

Contribution: The focus of this paper is the minmax regret path center problem on trees. For this problem, Puerto *et al.*'s algorithm requires $O(n^2)$ time. This paper presents an $O(n \log n)$-time algorithm. The bottleneck of Puerto *et al.*'s algorithm is to compute the classical path centers of the given tree under n different settings of node weights. For each setting, their algorithm finds the classical path center in $O(n)$ time. Our improvement is established on the following simple observation: the n settings of node weights are almost the same. Based on this observation, we preprocess the given tree in $O(n \log n)$ time to compute some useful auxiliary data structures; and then use the computed data structures to find the path center under each setting in $O(\log n)$ time.

Section 2 gives notation and definitions. Section 3 describes Puerto et al. algorithm [21] for finding a minmax regret path center of a tree. Section 4 presents efficient algorithms for a problem, called the entry vertex problem, and its extension. Then, using the algorithms in Sect. 4, Sect. 5 gives an improved $O(n \log n)$-time algorithm.

2 Notation and Definitions

Let $T = (V, E)$ be a tree, where V is the vertex set and E is the edge set. Let $n = |V|$. In this paper, T also denotes the set of all points of the tree. Thus, the notation $x \in T$ means that x is a point along any edge of T, which may or may not be a vertex of T. Each edge e has a nonnegative length. For any two points $p, q \in T$, let $P(p, q)$ be the unique path from p to q and $d(p, q)$ be its length. Throughout this paper, we assume that T has been preprocessed so that $d(p, q)$ can be answered in $O(1)$ time for any $p, q \in V$. This preprocessing requires $O(n)$ time [10]. For a subgraph X of T, the vertex set and edge set of X are, respectively, $V(X)$ and $E(X)$. For each vertex $v \in V$, the subgraph having a vertex set $\{v\}$ is simply denoted by v. For any vertex $v \in V$ and subgraph X of T, the *distance* from v to X, denoted by $d(v, X)$, is the shortest distance from v to any point of X (i.e., $d(v, X) = \min_{x \in X} d(v, x)$) and $close(v, X)$ is the vertex or point in X nearest to v. A path in T is called a *v-path*, where $v \in V$, if v is one of its endpoints.

Each vertex $v \in V$ is associated with an interval $[w_v^-, w_v^+]$, where $0 \leq w_v^- \leq w_v^+$. The *weight* of each vertex $v \in V$ can be any value in the interval $[w_v^-, w_v^+]$. Let Σ be the Cartesian product of intervals $[w_v^-, w_v^+]$, where $v \in V$. Any element $S \in \Sigma$ is called a *scenario* and represents a feasible assignment of weights to the vertices of T. 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.

Let $S \in \Sigma$ be a scenario. For any two subgraphs X and Y of T, the *eccentricity* from X to Y under the scenario S is $C^S(X, Y) = \max_{v \in V(X)} w_v^S\, d(v, Y)$, which is the maximum weighted distance from any vertex in X to Y according to the scenario S. A path H that minimizes $C^S(T, H)$ is called a *path center* of T under the scenario S. The finding of a path center of T under a fixed scenario S is called the *classical path center problem*. We use $\pi(S)$ to denote a path center of T under a scenario S.

For any path H in T, the *regret* of H with respect to a scenario $S \in \Sigma$ is $R^S(H) = C^S(T, H) - C^S(T, \pi(S))$ and the *maximum regret* of H is $R^*(H) = \max_{S \in \Sigma} R^S(H)$. The *minmax regret path center problem* is to determine a path H in T that minimizes $R^*(H)$. The determined path is called a *minmax regret path center*.

For ease of discussion, throughout this paper, we assume that each internal vertex of T has exactly three neighbors. In case this is not true, the given tree is transformed into an equivalent tree in linear time [13, 19]. Consider an internal vertex $v \in V$. There are three subtrees of T attached to v through the edges incident on v. For each $(u, v) \in E$, we denote by T_u^v the subtree of T attached to v through the edge (u, v), excluding this edge and the vertex v. We define the *subtrees* of a path H to be the subtrees T_j^i such that i is an internal node of H and j is the neighbor of i that is not on H. For any $p, q \in V$, define $subtree(p, q)$ to be the union of the subtrees of $P(p, q)$ (see Fig. 1). For ease of description, sometimes we will orient T into a rooted tree. In such a case, for each node $v \in V$, we use $p(v)$ and $sib(v)$ to denote, respectively, its parent and sibling, and use T_v to denote the subtree of T rooted at v.

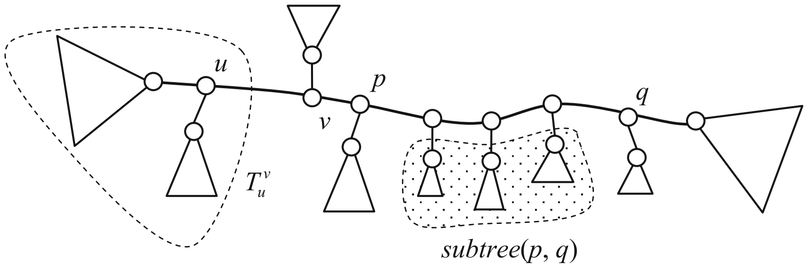

Fig. 1. Subtree T_u^v and $subtree(p, q)$.

3 Puerto, Ricca, and Scozzari's Algorithm

Puerto et al. [21] had an $O(n^2)$-time algorithm for finding a minmax regret path center of a tree. This section reviews their algorithm.

Recall that $\pi(S)$ denotes a path center of T under a scenario S. For any scenario $S \in \Sigma$, let $\alpha(S) = C^S(T, \pi(S))$. For each $i \in V$, let S_i be the scenario in which the weight of vertex i is w_i^+ and the weight of any other vertex v is w_v^-. Based on an augmented tree approach introduced by Averbakh and Berman [3], Puerto, Ricca, and Scozzari solved the minmax regret path center problem by an elegant transformation to the classical path center problem. Define an auxiliary tree T' as follows. Let M be a number that is larger than $\alpha(S_i)$ for any $i \in V$. The tree T' is obtained from T by appending to each vertex $i \in V$ a vertex i' and an edge (i, i') with length $(M - \alpha(S_i))/w_i^+$. Specific weights are assigned to the vertices of T'. For each $i \in V$, the weight of i is zero and the weight of i' is w_i^+. Let P be a path in the auxiliary tree T'. The *restriction* of P to T is the path obtained from P by deleting the edges of P that are not in T. Puerto, Ricca, and Scozzari gave the following nice property for solving the minmax regret path center problem.

Lemma 1 [21]. Let P be a path center of T'. Then, the restriction of P to T is a minmax regret path center of T.

Based upon Lemma 1, Puerto, Ricca, and Scozzari solved the minmax regret path center problem in $O(n^2)$ time as follows. First, $\alpha(S_i)$ is computed for each $i \in V$. By using the linear-time algorithm in [7] for the classical path center problem on a tree, this step is done in $O(n^2)$ time. Next, the auxiliary tree T' is constructed, which requires $O(n)$ time. Finally, a solution is obtained by applying the algorithm in [7] again to T'.

4 The Entry Vertex Problem

A *rooted path center* of a rooted tree with root r under a fixed scenario is an r-path H that minimizes the eccentricity from the tree to H. For a rooted path center, the endpoint other than the root is called its *terminal*, which may be a vertex or an interior point of an edge. A rooted tree may have more than one rooted path center. However, it is easy to see that the shortest one is unique and can be obtained as follows: initially set the terminal at the root and then continuously extend it toward a farthest vertex below it, until the extension does not decrease the eccentricity.

The entry vertex problem is defined as follows. Let $T = (V, E)$ be a tree with a fixed scenario S. For any $(u, v) \in E$, let $Q^S(T_u^v)$ be the *shortest* rooted path center of T_u^v under the scenario S, where T_u^v is considered as a rooted tree with root u. The *entry vertex* of a vertex x to a path H is the vertex on H nearest to x. For any $(u, v) \in E$ and $x \in V(T_u^v)$, define ENTRY(x, T_u^v) to be a query that returns the entry vertex of x to the shortest rooted path center, $Q^S(T_u^v)$, of T_u^v. (See Fig. 2) Note that ENTRY(x, T_u^v) may not be the same as *close*$(x, Q^S(T_u^v))$, since only vertices can be entry vertices. The *entry vertex problem* is to preprocess the tree T such that each ENTRY query can be answered efficiently.

This section shows that with an $O(n \log n)$-time preprocessing, each ENTRY query can be answered in $O(\log n)$ time.

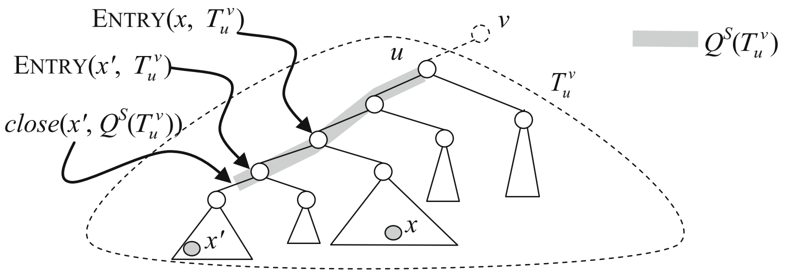

Fig. 2. Entry vertices to the shortest rooted path center of T_u^v.

4.1 Preprocessing

A query NODE(p, q, k), where $p, q \in V$ and k is an integer, requests the k-th vertex on the path from p to q. For any two vertices $p, q \in V$ and scenario $S \in \Sigma$, let $M^S(p, q) = C^S(\mathit{subtree}(p, q), P(p, q))$, which is the eccentricity of $P(p, q)$ from its subtrees. We need the following two lemmas.

Lemma 2. With an $O(n)$-time preprocessing, a query NODE(p, q, k) can be answered in $O(1)$ time for any $p, q \in V$ and integer k.

Lemma 3. Suppose that $C^S(T_u^v, v)$ of all $(u, v) \in E$ are given. Then, with an $O(n)$-time preprocessing, $M^S(p, q)$ can be computed in $O(1)$ time for any $p, q \in V$.

Given a rooted tree in which each node is associated with a *cost*, a query $lca(a, b)$ requests the least common ancestor of two vertices a, b; a query $la(v, l)$ requests the level l ancestor of a vertex v, where the level of a vertex is the number of edges from it to the root; and a query $\textsc{Max}(a, b)$ requests the largest cost of the vertices on a path $P(a, b)$. It was shown in [1, 13] that after an $O(n)$ time preprocessing, each lca, la, and $\textsc{Max}$ query can be answered in $O(1)$ time. Using these results, it is not difficult to prove the above two lemmas.

Using the divide-and-conquer approach, Tamir [25] gave an $O(n \log n)$-time algorithm to compute $C^S(T, v)$ for all $v \in V$. Based on the same idea and the top tree data structure in [2], we can show the following.

Lemma 4. The computation of $C^S(T_u^v, v)$ for all $(u, v) \in E$ can be done in $O(n \log n)$ time.

We proceed to describe the preprocessing algorithm. First, we compute $C^S(T_u^v, v)$ for every $(u, v) \in E$. By Lemma 4, this step requires $O(n \log n)$ time. Next, by Lemmas 2 and 3, we preprocess T so that $\textsc{Node}(p, q, k)$ and $M^S(p, q)$ can be obtained in $O(1)$ time for any $p, q \in V$ and integer k.

4.2 Algorithm for Queries

Consider a query $\textsc{Entry}(x, T_u^v)$. For notational simplicity, in this section, we assume that T_u^v is rooted at u. In addition, since the scenario S is fixed, we write $C(\cdot, \cdot)$, $M(\cdot, \cdot)$, and $Q(\cdot)$, respectively, for $C^S(\cdot, \cdot)$, $M^S(\cdot, \cdot)$, and $Q^S(\cdot)$. Our query algorithm finds the answer in a binary search manner, mainly based upon the following.

Lemma 5. A vertex i of T_u^v is on the path $Q(T_u^v)$ if and only if $C(T_i, i) \geq M(v, i)$.

Proof. Assume first that $C(T_i, i) \geq M(v, i)$. Consider an u-path H in T_u^v not containing i. Since $C(T_u^v, H) \geq C(T_i, H) > C(T_i, i)$ and $C(T_u^v, P(u, i)) = \max\{C(T_i, i), M(v, i)\} = C(T_i, i)$, we have $C(T_u^v, H) > C(T_u^v, P(u, i))$. Thus, any u-path in T_u^v not containing i is not a rooted path center. Therefore, the if-part holds.

Next, assume that i is a vertex on $Q(T_u^v)$. By contradiction, suppose that $C(T_i, i) < M(v, i)$. Since $Q(T_u^v)$ passes through i, we have $C(T_i, Q(T_u^v)) \leq C(T_i, i) < M(v, i)$. Therefore, $C(T_u^v, Q(T_u^v)) = \max\{C(T_i, Q(T_u^v)), M(v, i)\} = M(v, i)$. Let t be any point of edge $(i, p(i))$ such that $0 < d(i, t) \leq (M(v, i) - C(T_i, i))/w^*$, where w^* is the largest weight in T_i. Consider the path $P(u, t)$, which is shorter than $Q(T_u^v)$. Clearly, $C(T_i, t) \leq C(T_i, i) + d(i, t) \times w^* \leq M(v, i)$. Since $C(T_i, t) \leq M(v, i)$, we have $C(T_u^v, P(u, t)) = \max\{C(T_i, t), M(v, i)\} = M(v, i) = C(T_u^v, Q(T_u^v))$, which contradicts that $Q(T_u^v)$ is the shortest rooted path center of T_u^v. Therefore, $C(T_i, i) \geq M(v, i)$. Consequently, the lemma holds. □

The entry vertex $m = \textsc{Entry}(x, T_u^v)$ is found as follows. By definition, m is the first vertex on $P(x, u)$ that is contained in $Q(T_u^v)$. All successors of m on $P(x, u)$ are contained in $Q(T_u^v)$; and all predecessors of m on $P(x, u)$ are not contained in $Q(T_u^v)$. Therefore, m can be identified by performing binary search on $P(x, u)$. With the help of $\textsc{Node}$ queries, any node on $P(x, u)$ can be accessed in $O(1)$ time. By Lemma 5, whether a vertex i is on $Q(T_u^v)$ can be checked in $O(1)$ time by using the values of $C(T_i, i)$ and $M(v, i)$. After the preprocessing in Sect. 4.1, $C(T_i, i) = \max\{C(T_a^i, i), C(T_b^i, i)\}$ and $M(v,$

i) can be computed in $O(1)$ time for any vertex i in T_u^v, where a and b are the two children of i. Therefore, we have the following.

Theorem 1. With an $O(n \log n)$-time preprocessing, each Entry query can be answered in $O(\log n)$ time.

4.3 An Extended Problem

Our improvement on the minmax regret path center problem is based on solving an extended version of the entry vertex problem, in which it is allowed to temporarily increase the weight of a vertex during a query. For any $i \in V$ and $w \geq w_i^S$, we use $S|(i, w)$ to denote the scenario obtained from S by increasing the weight of i to w. A query ExtendEntry(x, T_u^v, i, w) reports the entry vertex of x to the rooted path center of T_u^v under the scenario $S|(i, w)$, where $(u, v) \in E$, $x \in V(T_u^v)$, $i \in V$, and $w \geq w_i^S$. That is, ExtendEntry(x, T_u^v, i, w) reports the entry vertex of x to the path $Q^{S|(i,w)}(T_u^v)$. In the following, we show that after an $O(n \log n)$-time preprocessing, each ExtendEntry query can also be answered in $O(\log n)$ time. Using the *lca* algorithm in [13], it is not difficult to prove the following lemma.

Lemma 6. With an $O(n)$-time preprocessing, $close(x, P(p, q))$ can be computed in $O(1)$ time for any three vertices $p, q, x \in V$.

Lemma 7. With an $O(n \log n)$-time preprocessing, $C^{S|(i,w)}(T_u^v, v)$ and $M^{S|(i,w)}(p, q)$ can be computed in $O(1)$ time for any $i, p, q \in V$, $(u, v) \in E$, and $w \geq w_i^S$.

Proof. As in Sect. 4.1, we preprocess T so that $C^S(T_u^v, v)$ for any $(u, v) \in E$ and $M^S(p, q)$ for any vertices $p, q \in V$ can be computed in $O(1)$ time. In addition, we preprocess T so that $close(x, P(p, q))$ can be accessed in $O(1)$ time for any $x, p, q \in V$.

For any $i \in V$, $(u, v) \in E$, and $w \geq w_i^S$, since $S|(i, w)$ differs from S only in the weight of i, $C^{S|(i,w)}(T_u^v, v)$ is computed in $O(1)$ time as follows. First, determine whether $i \in T_u^v$ by checking whether $close(i, P(u, v)) = u$. Next, if $i \in T_u^v$, we set $C^{S|(i,w)}(T_u^v, v) = \max\{C^S(T_u^v, v), w \times d(i, v)\}$; otherwise, we set $C^{S|(i,w)}(T_u^v, v) = C^S(T_u^v, v)$. For any $i, p, q \in V$, $M^{S|(i,w)}(p, q)$ is computed in $O(1)$ time as follows. First, determine whether i is a vertex in $subtree(p, q)$ or an internal node of $P(p, q)$ by checking whether $close(i, P(p, q)) \notin \{p, q\}$. Next, if i is a vertex in $subtree(p, q)$ or an internal node of $P(p, q)$, we set $M^{S|(i,w)}(p, q) = \max\{M^S(p, q), w \times d(i, close(i, P(p, q)))\}$; otherwise, we set $M^{S|(i,w)}(p, q) = M^S(p, q)$. Consequently, the lemma holds. □

Consider a query ExtendEntry(x, T_u^v, i, w). According to the query algorithm in Sect. 4.2, to show that this query can be answered in $O(\log n)$ time, it suffices to show that $C^{S|(i,w)}(T_u^v, v)$, $M^{S|(i,w)}(p, q)$, and Node(p, q, k) can be obtained in $O(1)$ time for any $i, p, q \in V$, $(u, v) \in E$, and integer k. As a result, by combining Lemmas 2 and 7, we obtain the following.

Theorem 2. Let T be a tree with a fixed scenario S. With an $O(n \log n)$-time preprocessing, each ExtendEntry query can be answered in $O(\log n)$ time.

5 An Improved Algorithm for the Path Center Problem

The bottleneck of the algorithm in [21] is to compute $\alpha(S_i)$ for every $i \in V$. Recall that for any scenario $S \in \Sigma$, $\alpha(S)$ denotes $C^S(T, \pi(S))$ and for each $i \in V$, S_i denotes the scenario in which the weight of vertex i is w_i^+ and the weight of any other vertex v is w_v^-. In this section, we improve the upper bound of the minmax regret path center problem on a tree by showing that the computation of all $\alpha(S_i)$ can be done in $O(n \log n)$ time.

Let S^- be the scenario in which the weight of every vertex v is w_v^-. The scenario S^- differs from each S_i only in the weight of vertex i. Our idea is to preprocess T under the scenario S^-, so that each $\alpha(S_i)$ can be determined efficiently. Let $M^S(p, q)$ and $Q^S(T_u^v)$ be defined the same as in Sect. 4. For any $(u, v) \in E$ and scenario S, let $\lambda^S(T_u^v) = C^S(T_u^v, Q^S(T_u^v))$, which is the eccentricity from T_u^v to the rooted path center $Q^S(T_u^v)$. For notational simplicity, in this section, $C^{S^-}(\cdot,\cdot), C^{S_i}(\cdot,\cdot), M^{S^-}(\cdot,\cdot), M^{S_i}(\cdot,\cdot), l^{S^-}(\cdot)$, and $\lambda^{S_i}(\cdot)$ are simply denoted, respectively, by $C^-(\cdot, \cdot)$, $C^i(\cdot, \cdot)$, $M^-(\cdot, \cdot)$, $M^i(\cdot, \cdot)$, $\lambda^-(\cdot)$, and $\lambda^i(\cdot)$.

Lemma 8. Suppose that $C^-(T_u^v, v)$ for all $(u, v) \in E$ are given. In $O(n)$ time, we can compute $\lambda^-(T_u^v)$ for all edges $(u, v) \in E$.

Proof. In this proof, we assume that T is under the scenario S^-. We orient T into a rooted tree with an arbitrary root r. Since there always exists a rooted path center whose terminal is a leaf, using the dynamic programming approach, all $\lambda^-(T_u^v)$ are computed in two phases.

Phase 1. This phase computes $\lambda^-(T_x)$ for all $x \in V$ in a bottom-up manner as follows. If x is a leaf, we have $\lambda^-(T_x) = 0$. Assume that x is an internal vertex and let x_1, x_2 be its two children. Let H be a rooted path center of T_x. If H passes through x_1, since $\lambda^-(T_x) = C^-(T_x, H) = \max\{C^-(T_{x1}, H), C^-(T_{x2}, x)\}$ and a rooted path center of T_{x1} has the minimum eccentricity from T_{x1} among all x_1-paths, it can be concluded that $\lambda^-(T_x) = \max\{\lambda^-(T_{x1}), C^-(T_{x2}, x)\}$. Similarly, if H passes through x_2, it can be concluded that $\lambda^-(T_x) = \max\{C^-(T_{x1}, x), \lambda^-(T_{x2})\}$. Therefore, we compute $\lambda^-(T_x)$ as $\min\{\max\{\lambda^-(T_{x1}), C^-(T_{x2}, x)\}, \max\{C^-(T_{x1}, x), \lambda^-(T_{x2})\}\}$.

Phase 2. This phase computes $\lambda^-(T_{p(x)}^x)$ for all $x \in V$ in a top-down manner as follows. If x is the root r, we have $\lambda^-(T_{p(x)}^x) = \lambda^-(\emptyset) = 0$. Assume that $x \neq r$. A rooted path center of $T_{p(x)}^x$ passes through either $sib(x)$ or $p(p(x))$. If it passes through $sib(x)$, we have $\lambda^-(T_{p(x)}^x) = \max\{\lambda^-(T_{sib(x)}), C^-(T_{p(p(x))}^{p(x)}, p(x))\}$; otherwise, we have $\lambda^-(T_{p(x)}^x) = \max\{C^-(T_{sib(x)}, p(x)), \lambda^-(T_{p(p(x))}^{p(x)}, p(x))\}$. Therefore, $\lambda^-(T_{p(x)}^x)$ can be computed in $O(1)$ time.

The above computation requires $O(n)$ time. Thus, the lemma holds. □

Lemma 9. Suppose that the following can be accessed in $O(1)$ time: $\lambda^{-}(T_u^v)$ for any $(u, v) \in E$, $C^i(T_u^v, v)$ for any $(u, v) \in E$ and $i \in V$, and $M^i(p, q)$ for any $i, p, q \in V$; and suppose that the entry vertex of i to $Q^{S_i}(T_u^v)$ can be accessed in $O(\log n)$ time for any $(u, v) \in E$ and $i \in V$. Then, $\lambda^i(T_u^v)$ can be computed in $O(\log n)$ time for any $(u, v) \in E$ and $i \in V$.

Proof. We prove this lemma by presenting an algorithm. For ease of description, assume that T_u^v is rooted at u and is under the scenario S_i. First, compute m as the entry vertex of i to $Q^{S_i}(T_u^v)$ in $O(\log n)$ time. Let m_1, m_2 be the two children of m. Next, in $O(1)$ time, we find the values of $C^i(T_{m1}, m)$ and $C^i(T_{m2}, m)$. By symmetry, assume that $C^i(T_{m1}, m) \geq C^i(T_{m2}, m)$. We first establish the following claim.

Claim. There is a rooted path center of T_u^v (under S_i) that passes through m_1.

Proof of the Claim. Let $P(u,t) = Q^{S_i}(T_u^v)$. Clearly, any u-path containing $Q^{S_i}(T_u^v)$ is a rooted path center. Thus, to prove this claim, we only need to show that the terminal t is a point of edge (m, m_1) or is in T_{m1}. Since m is a vertex on $Q^{S_i}(T_u^v)$, $Q^{S_i}(T_u^v)$ can be obtained by initially setting the terminal t at m and then continuously extending it toward a farthest vertex below it, until the extension does not decrease the eccentricity. Since $C^i(T_{m1}, m) \geq C^i(T_{m2}, m)$, it is easy to conclude that at $t = m$ an extension can decrease the eccentricity only if it is toward the vertex m_1. Therefore, t is a point of edge (m, m_1) or is in T_{m1}. Consequently, the claim follows.

We now complete the proof of the lemma. Let H be a rooted path center of T_u^v that passes through m_1. Two cases are discussed.

Case 1: $i \in V(T_{m1})$.
In this case, m_1 is not on the shortest path center $Q^{S_i}(T_u^v)$. Otherwise, since m_1 is closer to x than m, m is not the entry vertex of i to $Q^{S_i}(T_u^v)$. By Lemma 4, $C^i(T_{m1}, m_1) < M^i(v, m_1)$. Since H passes through m_1, we have $C^i(T_{m1}, H) \leq C^i(T_{m1}, m_1) < M^i(v, m_1)$. Therefore, $\lambda^i(T_u^v) = C^i(T_u^v, H) = \max\{C^i(T_{m1}, H), C^i(subtree(v, m_1), H\} = \max\{C^i(T_{m1}, H), M^i(v, m_1)\} = M^i(v, m_1)$. Consequently, in this case, we compute $\lambda^i(T_u^v) = M^i(v, m_1)$ in $O(1)$ time.

Case 2: $i \notin V(T_{m1})$.
Since $i \notin V(T_{m1})$, we have $C^i(T_{m1}, H) = C^{-}(T_{m1}, H)$ and thus $C^i(T_u^v, H) = \max\{C^{-}(T_{m1}, H), M^i(v, m_1)\}$. Let H^* be the union of $P(u, m_1)$ and $Q^{S^-}(T_{m1})$. Under S^-, the path $Q^{S^-}(T_{m1})$ has the minimum eccentricity from T_{m1} among all m_1-paths in T_{m1}. Consequently, it can be concluded that $C^i(T_u^v, H^*) = \max\{C^{-}(T_{m1}, H^*), M^i(v, m_1)\} \leq C^i(T_u^v, H)$. Therefore, H^* is also a rooted path center of T under S_i and thus $\lambda^i(T_u^v) = \max\{C^{-}(T_{m1}, H^*), M^i(v, m_1)\} = \max\{\lambda^{-}(T_{m1}), M^i(v, m_1)\}$. Consequently, in this case, we compute $\lambda^i(T_u^v) = \max\{\lambda^{-}(T_{m1}), M^i(v, m_1)\}$ in $O(1)$ time.

The above computation of $\lambda^i(T_u^v)$ requires $O(\log n)$ time. Thus, the lemma holds. □

A *discrete 1-center* of T under a scenario S is a vertex $v \in V$ that minimizes $C^S(T, v)$. Tamir *et al.* [26] gave the following.

Lemma 10 [26]. Let T be a tree with a fixed scenario and c be its discrete 1-center. Then, T has a path center that contains c.

We proceed to present an algorithm for computing $\alpha(S_i)$ of all $i \in V$. Consider the computation for a fixed $i \in V$. Assume that T is under the scenario S_i. Let c be a discrete 1-center of T. By Lemma 10, there is a path center containing c. Let x, y, z be the three children of c. Without losing any generality, assume that $C^i(T^c_x, c) \geq C^i(T^c_y, c) \geq C^i(T^c_z, c)$. Then, there exists a path center that passes through x and y [26]. For any path H passing through x and y, we have $C^i(T, H) = \max\{C^i(T^c_x, H), C^i(T^c_y, H), C^i(T^c_z, c)\}$. Let H^* be the union of $Q^{S_i}(T^c_x)$, $P(x, y)$, and $Q^{S_i}(T^c_y)$. The path $Q^{S_i}(T^c_x)$ has the minimum eccentricity from T^c_x among all x-paths in T^c_x; and the path $Q^{S_i}(T^c_y)$ has the minimum eccentricity from T^c_y among all y-paths in T^c_y. Consequently, it can be concluded that H^* has the minimum eccentricity among all paths that pass through x and y. That is, H^* is a path center of T. Therefore, $\alpha(S_i)$ can be computed as $C^i(T, H^*) = \max\{C^i(T^c_x, Q^{S_i}(T^c_x)), C^i(T^c_y, Q^{S_i}(T^c_y)), C^i(T^c_z, c)\} = \max\{\lambda^i(T^c_x), \lambda^i(T^c_y), C^i(T^c_z, c)\}$.

Based upon the above discussion, an algorithm for computing all $\alpha(S_i)$ is described as follows.

```
Algorithm 1. ALLPATHCENTERS
Input: a tree $T = (V, E)$ and $[w_i^-, w_i^+]$ for each vertex $i \in V$
Output: $\alpha(S_i)$ for each vertex $i \in V$
begin
1.  preprocess $T$ so that $C^-(T_u^v, v)$, $C^i(T_u^v, v)$, and $M^i(p, q)$ can be obtained in $O(1)$ time
    for any $(u, v) \in E$ and $i, p, q \in V$
2.  preprocess $T$ so that the entry vertex of $x$ to $Q^{S_i}(T_u^v)$ can be determined in $O(\log n)$
    time for any $x, i \in V$ and $(u, v) \in E$
3.  compute $\lambda^-(T_u^v)$ for every $(u, v) \in E$
4.  compute the discrete 1-center of $T$ under $S_i$ for each $i \in V$
5.  for each vertex $i \in V$ do          /* compute $\alpha(S_i)$
6.  begin
7       $c \leftarrow$ the discrete 1-center computed for $S_i$ in Line 4
8       $(x, y, z) \leftarrow$ the three neighbors of $c$, where $C^i(T_x^c, c) \geq C^i(T_y^c, c) \geq C^i(T_z^c, c)$
9       compute $\lambda^i(T_x^c)$ and $\lambda^i(T_y^c)$
10      $\alpha(S_i) \leftarrow \max\{\lambda^i(T_x^c), \lambda^i(T_y^c), C^i(T_z^c, c)\}$
11  end
12  return $(\{\alpha(S_i) \mid i \in V\})$
end
```

Since $S_i = S^-|(i, w_i^+)$ for each $i \in V$, by using Lemmas 4 and 7 with $S = S^-$, Line 1 requires $O(n \log n)$ time. By definition, when T is under S^-, the entry vertex of x to $Q^{S_i}(T^v_u)$ is EXTENDENTRY(x, T^v_u, i, w_i^+). Therefore, by using Theorem 2 with $S = S^-$, Line 2 requires $O(n \log n)$ time. By Lemma 8, Line 3 takes $O(n)$ time. Yu et al. [31] showed that a discrete 1-center of T under S_i can be computed in $O(n \log n)$ time for every $i \in V$. Thus, Line 4 requires $O(n \log n)$ time. Consider the for-loop in Lines 5–11. Lines 7, 8, 10 take $O(1)$ time. By Lemma 9, after the preprocessing in Lines 1, 2, and 3, the computation of $\lambda^i(T^c_x)$ and $\lambda^i(T^c_y)$ in Line 9 can be done in $O(\log n)$ time. Therefore, each iteration of the for-loop requires $O(\log n)$ time. As a result, we obtain the following.

Lemma 11. We can compute $\alpha(S_i)$ for all $i \in V$ in $O(n \log n)$ time.

Theorem 3. The minmax regret path center problem on a tree can be solved in $O(n \log n)$ time.

References

1. Alstrup, S., Holm, J.: Improved algorithms for finding level ancestors in dynamic trees. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 73–84. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45022-X_8
2. Alstrup, S., Lauridsen, Peter W., Sommerlund, P., Thorup, M.: Finding cores of limited length. In: Dehne, F., Rau-Chaplin, A., Sack, J.-R., Tamassia, R. (eds.) WADS 1997. LNCS, vol. 1272, pp. 45–54. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63307-3_47
3. Averbakh, I., Berman, O.: Minimax regret p-center location on a network with demand uncertainty. Locat. Sci. **5**(4), 247–254 (1997)
4. Averbakh, I., Berman, O.: Minmax regret median location on a network under uncertainty. INFORMS J. Comput. **12**(2), 104–110 (2000)
5. Bhattacharya, B., Kameda, T., Song, Z.: A linear time algorithm for computing minmax regret 1-median on a tree network. Algorithmica **70**(1), 2–21 (2014)
6. Bhattacharya, B., Kameda, T., Song, Z.: Minmax regret 1-center algorithms for path/tree/unicycle/cactus networks. Discrete Appl. Math. **195**, 18–30 (2015)
7. Bhattacharya, B., Shi, Q., Tamir, A.: Optimal algorithms for the path/tree-shaped facility location problems in trees. Algorithmica **55**(4), 601–618 (2009)
8. Conde, E.: Minmax regret location–allocation problem on a network under uncertainty. Eur. J. Oper. Res. **179**(3), 1025–1039 (2007)
9. Conde, E.: A note on the minmax regret centdian location on trees. Oper. Res. Lett. **36**(2), 271–275 (2008)
10. Djidjev, H.N., Pantziou, G.E., Zaroliagis, C.D.: Computing shortest paths and distances in planar graphs. In: Albert, J.L., Monien, B., Artalejo, M.R. (eds.) ICALP 1991. LNCS, vol. 510, pp. 327–338. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-54233-7_145
11. Drezner, Z.: Sensitivity analysis of the optimal location of a facility. Naval Res. Logist. Q. **32**(2), 209–224 (1985)
12. Frank, H.: Optimum locations on a graph with probabilistic demands. Oper. Res. **14**(3), 409–421 (1966)
13. Harel, D., Tarjan, R.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. **13**(2), 338–355 (1984)
14. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems. I: the p-centers. SIAM J. Appl. Math. **37**(3), 513–538 (1979)
15. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems. II: the p-medians. SIAM J. Appl. Math. **37**(3), 539–560 (1979)
16. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Springer, New York (2013). https://doi.org/10.1007/978-1-4757-2620-6
17. Labbé, M., Thisse, J.-F., Wendell, R.E.: Sensitivity analysis in minisum facility location problems. Oper. Res. **39**(6), 961–969 (1991)
18. Minieka, E.: The optimal location of a path or tree in a tree network. Networks **15**(3), 309–321 (1985)
19. Mirchandani, P.B., Odoni, A.R.: Locations of medians on stochastic networks. Transp. Sci. **13**(2), 85–97 (1979)

20. Morgan, C.A., Slater, P.J.: A linear algorithm for a core of a tree. J. Algorithms **1**(3), 247–258 (1980)
21. Puerto, J., Ricca, F., Scozzari, A.: Minimax regret path location on trees. Networks **58**(2), 147–158 (2011)
22. Puerto, J., Rodríguez-Chía, A.M., Tamir, A., Pérez-Brito, D.: The bi-criteria doubly weighted center-median path problem on a tree. Networks **47**(4), 237–247 (2006)
23. Synder, L.: Facility location under uncertainty: a review. IIE Trans. **38**(7), 547–564 (2006)
24. Tamir, A.: An O(pn^2) algorithm for the p-median and related problems on tree graphs. Oper. Res. Lett. **19**(2), 59–64 (1996)
25. Tamir, A.: Sorting weighted distances with applications to objective function evaluations in single facility location problems. Oper. Res. Lett. **32**(3), 249–257 (2004)
26. Tamir, A., Puerto, J., Mesa, J.A., Rodríguez-Chía, A.M.: Conditional location of path and tree shaped facilities on trees. J. Algorithms **56**(1), 50–75 (2005)
27. Tamir, A., Puerto, J., Pérez-Brito, D.: The centdian subtree on tree networks. Discrete Appl. Math. **118**(3), 263–278 (2002)
28. Wang, B.-F.: Efficient parallel algorithms for optimally locating a path and a tree of a specified length in a weighted tree network. J. Algorithms **34**(1), 90–108 (2000)
29. Ye, J.-H., Li, C.-Y., Wang, B.-F.: An improved algorithm for the minmax regret path centdian problem on trees. Submitted to journal for publication, under revision
30. Ye, J.-H., Wang, B.-F.: On the minmax regret path median problem on trees. J. Comput. Syst. Sci. **81**(7), 1159–1170 (2015)
31. Yu, H.-I., Lin, T.-C., Wang, B.-F.: Improved algorithms for the minmax-regret 1-center and 1-median problems. ACM Trans. Algorithms **4**(3), 1–27 (2008)