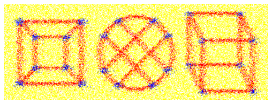

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 18, no. 4, pp. 539–555 (2014)
DOI: 10.7155/jgaa.00336

Minimax Regret Sink Location Problem in Dynamic Tree Networks with Uniform Capacity

Yuya Higashikawa[1] *Mordecai J. Golin*[2] *Naoki Katoh*[1]

[1]Department of Architecture and Architectural Engineering,
Kyoto University, Japan
[2]Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology, Hong Kong

Abstract

This paper addresses the minimax regret sink location problem in dynamic tree networks. In our model, a dynamic tree network consists of an undirected tree with positive edge lengths and uniform edge capacity, and the vertex supply which is a positive value is unknown but only the interval of supply is known. A particular realization of supply to each vertex is called a scenario. Under any scenario, the cost of a sink location x is defined as the minimum time to complete the evacuation to x for all supplies (evacuees), and the regret of x is defined as the cost of x minus the cost of the optimal sink location. Then, the problem is to find a sink location which minimizes the maximum regret for all possible scenarios. We present an $O(n^2 \log^2 n)$ time algorithm for the minimax regret sink location problem in dynamic tree networks with uniform capacity, where n is the number of vertices in the network. As a preliminary step for this result, we also address the minimum cost sink location problem in a dynamic tree networks under a fixed scenario and present an $O(n \log n)$ time algorithm, which improves upon the existing time bound of $O(n \log^2 n)$ by [14] if edges of a tree have uniform capacity.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
March 2014	July 2014	July 2014	July 2014	December 2014

Published:
December 2014

Article type: Regular paper
Communicated by: S. Pal and K. Sadakane

This research is supported by JSPS Grant-in-Aid for JSPS Fellows (26 · 4042) and JSPS Grant-in-Aid for Scientific Research(A)(25240004).

E-mail addresses: as.higashikawa@archi.kyoto-u.ac.jp (Yuya Higashikawa) golin@cs.ust.hk (Mordecai J. Golin) naoki@archi.kyoto-u.ac.jp (Naoki Katoh)

1 Introduction

Recently, big earthquakes frequently occur around the world, e.g., *the Great Hanshin-Awaji Earthquake* in 1995, *the Sichuan Earthquake* in 2008, and so on. Also, some big earthquakes triggers devastating tsunamis, e.g., *the Sumatra-Andaman Earthquake* in 2004 and *The Tohoku-Pacific Ocean Earthquake* in 2011. In such disasters, many people failed to evacuate and lost their lives due to severe attack by tsunamis. Moreover, not only earthquakes but also diverse disasters occur and cause serious damages in many countries. Therefore, from the viewpoint of disaster prevention, it has now become extremely important to establish effective evacuation planning systems against large scale disasters. In particular, arrangements of tsunami evacuation buildings in large Japanese cities near the coast have become an urgent issue. To determine appropriate tsunami evacuation buildings, we need to consider where evacuation buildings are located and how to partition a large area into small regions so that one evacuation building is designated in each region. This produces several theoretical issues to be considered. Among them, this paper focuses on the location problem of the evacuation building assuming that we fix the region such that all evacuees in the region are planned to evacuate to this building. A natural evaluation criterion of the building location is the time required to complete the evacuation. In order to treat this criterion, we consider the *dynamic* setting in graph networks, which was first introduced by Ford et al. [9]. Under the dynamic setting, each edge of a given graph has a capacity which limits the value of the flow into the edge at each time step. We call such networks under the dynamic setting *dynamic networks*.

This paper addresses *the minimax regret sink location problem in dynamic networks*. In our model, a dynamic network consists of an undirected graph with positive edge lengths and uniform edge capacity, and the vertex supply which is a positive value is unknown but only the interval of supply is known. Generally, the number of evacuees in an area (the initial supply at a vertex) may vary depending on the time (e.g., in an office area in a big city there are many people during the daytime on weekdays while there are much less people on weekends or during the night time). So, in order to take into account the uncertainty of the vertex supplies, we adopt the *maximum regret* for a particular sink location as another evaluation criterion assuming that we only know the interval of supply for each vertex. A particular realization (assignment of supply to each vertex) is called a *scenario*. Under any scenario, the *cost* of a sink location x is defined as the minimum time to complete the evacuation to x for all supplies, and the *regret* of x is defined as the cost of x minus the cost of the optimal sink location. Then, the problem can be understood as a 2-person Stackelberg game as follows. The first player picks a sink location x and the second player chooses a scenario s that maximizes the regret of x under s. The objective of the first player is to choose x that minimizes the maximum regret.

Several researchers have studied the minimax regret sink location problems [7, 13, 15, 16]. Especially, for tree networks, some efficient algorithms have been presented by [1, 2, 3, 4, 5, 8]. For dynamic networks, Cheng et

al. [6] have studied the minimax regret sink location problem in dynamic path networks with uniform capacity and presented an $O(n \log^2 n)$ time algorithm. Recently, Higashikawa et al. [10] improved the time bound by [6] to $O(n \log n)$. Also, Wang [17] independently achieved the same time bound of $O(n \log n)$ with better space complexity.

In this paper, we consider the minimax regret sink location problem in dynamic tree networks. There are both theoretical and practical motivations to study this problem. For the theoretical motivation, we are interested in how we can extend the solvable class of networks for the problem from dynamic path networks by [6, 10, 17]. In fact, the minimax regret sink location problem in dynamic tree networks has not been studied in the literature. For the practical motivation, as mentioned in [14], considering tree networks seems to be important since one of the desirable evacuation plans in a region sends evacuees to the evacuation building so that any evacuees never cross each other.

For the minimax regret 1-sink location problem in dynamic tree networks with uniform capacity, we propose an $O(n^2 \log^2 n)$ time algorithm. In order to develop this algorithm, we first consider the case where the supply at each vertex is fixed to a given value. The problem is to find a sink location in a given tree which minimizes the time to complete the evacuation to the sink for all supplies under a fixed scenario, which is called *the minimum cost sink location problem in dynamic tree networks*. An algorithm for this problem can be used as a subroutine of the algorithm to solve the minimax regret sink location problem in dynamic tree networks. Mamada et al. [14] have studied the minimum cost sink location problem in dynamic tree networks with general capacity and presented an $O(n \log^2 n)$ time algorithm. In this paper, we present an $O(n \log n)$ time algorithm for the minimum cost sink location problem in dynamic tree networks with uniform capacity. Note that the paper by [14] assumed that a sink is located at a vertex while our paper assumes that a sink can be located at any point in the network.

There are two key ideas of the proposed algorithm to find the minimax regret sink location. The first one is that for a fixed vertex, we can identify $O(n)$ scenarios among which the worst case scenario exists when the sink is located at the vertex. The second one is that we repeatedly halve the size of the area where the optimal sink location exists. In Section 2, we will treat the minimum cost sink location problem in dynamic tree networks. In Section 3, we will treat the minimax regret sink location problem in dynamic tree networks. Note that all claims, lemmas and the main theorem in Section 2 indeed hold even if the edge capacity is uniform with an arbitrary value, Lemma 4 and the main theorem in Section 3 holds only if the edge capacity is uniformly 1.

2 Minimum cost sink location problem in dynamic tree networks with uniform capacity

Let $T = (V, E)$ be an undirected tree with a vertex set V and an edge set E. Let $\mathcal{N} = (T, l, w, c, \tau)$ be a dynamic network with the underlying graph being a tree T, where l is a function that associates each edge $e \in E$ with a positive integral length $l(e)$, w is also a function that associates each vertex $v \in V$ with a positive integral supply $w(v)$ representing the number of evacuees at v, c is a positive integral constant representing the capacity of each edge: the least upper bound for the number of the evacuees entering an edge per unit time, and τ is also a constant representing the time required for traversing the unit distance of each evacuee. We call such networks with tree structures *dynamic tree networks*.

2.1 Formula for the minimum completion time of the evacuation

In the following, for two integers i and j, let $[i, j] = \{k \in \mathbb{Z} \mid i \leq k \leq j\}$. We first show a formula representing the minimum completion time for the evacuation in a dynamic tree network with uniform capacity. In the following, we also use the notation T to denote the set of all points on edges in E including all vertices in V. For two points $p, q \in T$, let $d(p, q)$ denote the distance between p and q in T. More precisely, when p (resp. q) divides an edge $e_p = (u_p, v_p)$ (resp. $e_q = (u_q, v_q)$) with the ratio of λ_p to $1 - \lambda_p$ with $0 \leq \lambda_p \leq 1$ (resp. λ_q to $1 - \lambda_q$ with $0 \leq \lambda_q \leq 1$) and the shortest path from u_p to u_q in T includes p and q in this order, we define $d(p, q)$ as follows:

$$d(p, q) = \sum \{l(e) \mid e \text{ is on the shortest path from } u_p \text{ to } u_q \text{ in } T\} \\ -\lambda_p l(e_p) - \lambda_q l(e_q). \quad (1)$$

For a vertex $v \in V$, let $\delta(v)$ denote the set of vertices adjacent to v, and for a point $p \in T$ which is not at a vertex but on an edge $(u, v) \in E$, let $\delta(p)$ denote the set of two vertices u and v. For a sink location x given at a point in T, let $\Theta(x)$ denote the minimum time required for all evacuees on T to complete the evacuation to x. In this paper, we assume that for any vertex $v \in V$, any number of evacuees can stay at v, and if the sink is located at v, all evacuees on v can instantly finish their evacuation. Let us consider a point $p \in T$. If p is not at a vertex but on an edge $(u, v) \in E$, let us split (u, v) into two new edges (u, p) and (p, v) and regard p as a new vertex of T. Then, let $T(p)$ be a rooted tree made from T such that each edge has a natural orientation towards the root p, and for any vertex $v \in V$, let $T(p, v)$ be the subtree of $T(p)$ rooted at v. For a sink location x given at a point in T, let $\Theta(x, v)$ denote the minimum time required for all evacuees on $T(x, v)$ to complete the evacuation to x. Then, we clearly have

$$\Theta(x) \quad = \quad \max\{\Theta(x, u) \mid u \in \delta(x)\}. \quad (2)$$

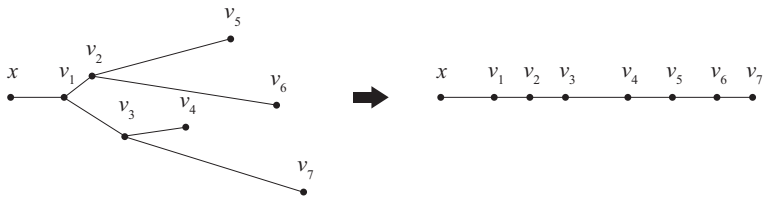

Figure 1: Vertices of the tree can be relocated on a line with the same edge capacities

Here, we only need to consider $\Theta(x, \hat{u})$ for $\hat{u} = \text{argmax}\{\Theta(x, u) \mid u \in \delta(x)\}$. Suppose that there are n' vertices in $T(x, \hat{u})$ named $v_1(= \hat{u}), v_2, \ldots, v_{n'}$ such that $d(x, v_i) \leq d(x, v_{i+1})$ for $i \in [1, n' - 1]$. Then, Kamiyama et al. [11] have observed that the value of $\Theta(x, \hat{u})$ does not change if x and all v_i for $i \in [1, n']$ are relocated on a line with the same edge capacities so that $d(x, v_i)$ for $i \in [1, n']$ remain the same (see Fig. 1), and $\Theta(x, \hat{u})$ can be represented as follows:

$$\Theta(x, \hat{u}) = \max_{j \in [1, n']} \left\{ d(x, v_j)\tau + \left\lceil \frac{\sum_{i \in [j, n']} w(v_i)}{c} \right\rceil - 1 \right\}. \tag{3}$$

For the completeness, we now see why this formula holds. We first define a *group* as a set of evacuees who simultaneously reach x from $\hat{u}$ and the *size* of a group as the number of evacuees in the group. Suppose that a group whose size is less than c reaches x at time t'. Then, we call a group which first reaches x after t' a *leading group* (see Fig. 2). We also call a group which first reaches x after time 0 a leading group. Let t_{last} denote the time when the last group reaches x (i.e., the whole evacuation finishes at t_{last}). Suppose that a leading group reaches x at time t'' and there is no leading group which reaches x after t'' until t_{last}. Then, we call the leading group reaching x at t'' the *last leading group* and a set of groups reaching x from t'' to t_{last} the *last cluster*. In order to derive $\Theta(x, \hat{u})$, we only need to observe the last cluster. We notice that all evacuees of a leading group are located at vertices whose distance from x are the same at time 0, and they all reach x without being blocked. Suppose that all evacuees of the last leading group are located at vertices $v_l, v_{l+1}, \ldots, v_{l+k}$ such that $d(x, v_l) = d(x, v_{l+1}) = \ldots = d(x, v_{l+k})$ at time 0. Then, the last leading group reaches x at time $d(x, v_l)\tau$, and then, all groups except ones which belong to the last cluster have already reached x. If $d(x, v_l)\tau < t_{\text{last}}$, the size of a group

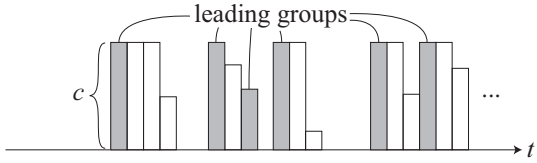

Figure 2: The size of groups reaching x from $\hat{u}$ for each time

reaching x at each time $t \in [d(x, v_l)\tau, t_{\text{last}} - 1]$ is exactly c by the definition of the last leading group. Therefore, $\Theta(x, \hat{u})$ can be represented as follows:

$$\Theta(x, \hat{u}) = d(x, v_l)\tau + \left\lceil \frac{\sum_{i \in [l, n']} w(v_i)}{c} \right\rceil - 1. \tag{4}$$

Note that this still holds for the case of $d(x, v_l)\tau = t_{\text{last}}$. We next see that the right hand of the formula (3) is the lower bound for $\Theta(x, \hat{u})$. For all evacuees located at $v_j, \ldots, v_{n'}$ with some integer $j \in [1, n']$, the time of $d(x, v_j)\tau + \lceil \sum_{i \in [j, n']} w(v_i)/c \rceil - 1$ is at least required to complete the evacuation to x, thus we have $\Theta(x, \hat{u}) \geq d(x, v_j)\tau + \lceil \sum_{i \in [j, n']} w(v_i)/c \rceil - 1$ for any integer $j \in [1, n']$. From the above discussion, we can derive the formula (3).

2.2 Properties

In this section, we prove the two lemmas which are key to our algorithm. Let x_{opt} denote a point in T which minimizes $\Theta(x)$. For two vertices $v, v' \in V$, let $V(v, v')$ denote the set of all vertices in $T(v, v')$ and $T(V')$ denote a subgraph induced by a vertex set $V' \subseteq V$.

Lemma 1 *Along a path from a leaf to another leaf in T, function $\Theta(x)$ is unimodal in x.*

Lemma 2 *For a vertex $v \in V$, if $\hat{u} = \text{argmax}\{\Theta(v, u) \mid u \in \delta(v)\}$ holds, there exists $x_{\text{opt}} \in T(V(v, \hat{u}) \cup \{v\})$.*

In the proofs of these two lemmas, we use the following notations. Let P be a simple path in T from a leaf to another leaf, which is represented as the sequence of vertices $v_0, v_1, \ldots, v_k$ where v_0 and v_k are leaves. In the following, for a point $p \in P$, we abuse the notation p to denote $d(v_0, p)$. For a point $p \in P$, we call the direction to v_0 (resp. v_k) from p the *left direction* (resp. *right direction*). If a sink location x is given at a vertex v_i with $i \in [1, k]$ (resp. $[0, k-1]$), let $\Theta_L(x; P)$ (resp. $\Theta_R(x; P)$) denote the minimum time required to complete the evacuation to x for all evacuees on $T(x, v_{i-1})$ (resp. $T(x, v_{i+1})$). If x is given on an edge (v_i, v_{i+1}) with $i \in [0, k-1]$, let $\Theta_L(x; P)$ (resp. $\Theta_R(x; P)$) denote the minimum time required to complete the evacuation to x for all evacuees on $T(x, v_i)$ (resp. $T(x, v_{i+1})$). Also, for a vertex v_i with $i \in [1, k-1]$, let

$$\Theta_L^{+0}(v_i; P) = \lim_{\epsilon \to +0} \{\Theta_L(v_i + \epsilon; P)\}, \tag{5}$$

$$\Theta_R^{-0}(v_i; P) = \lim_{\epsilon \to +0} \{\Theta_R(v_i - \epsilon; P)\}. \tag{6}$$

We first show the following claim.

Claim 1 *Along a path P, function $\Theta_L(x; P)$ is increasing in x and function $\Theta_R(x; P)$ is decreasing in x.*

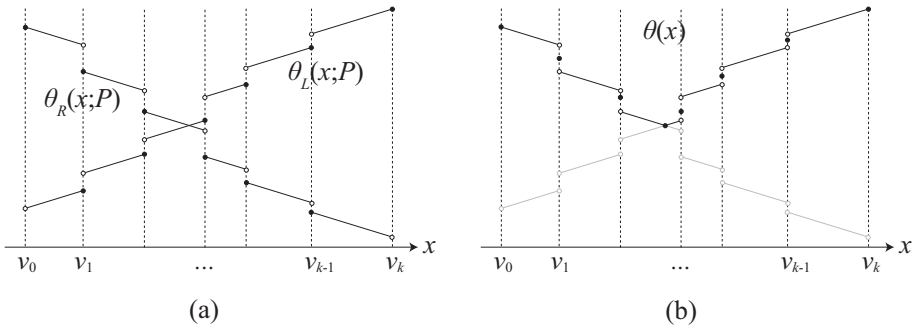

Figure 3: Functions along P: **(a)** $\Theta_L(x;P)$, $\Theta_R(x;P)$ and **(b)** $\Theta(x)$

Proof: By (3), (5) and (6), we can see the following three properties of $\Theta_L(x;P)$ and $\Theta_R(x;P)$ (see Fig. 3(a)): (i) for an open interval (v_{i-1}, v_i) with $i \in [1,k]$, $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is linear in x with slope τ (resp. $-\tau$), (ii) $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is left-continuous (resp. right-continuous) at $x = v_i$ for $i \in [1,k]$ (resp. $i \in [0,k-1]$), (iii) $\Theta_L(v_i;P) \leq \Theta_L^{+0}(v_i;P)$ (resp. $\Theta_R^{-0}(v_i;P) \geq \Theta_R(v_i;P)$) holds at v_i for $i \in [1,k-1]$. From these properties, $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is piecewise linear increasing (resp. decreasing) in x. □

By Claim 1, there uniquely exists $x \in P$ which minimizes $\max\{\Theta_L(x;P), \Theta_R(x;P)\}$, called $x_{\rm opt}(P)$ in the following. Then, we have the following claim.

Claim 2 (i) *For a vertex $v_i \in P$ such that $v_i \geq x_{\rm opt}(P)$, $\Theta_L(v_i;P) \leq \Theta(v_i) \leq \Theta_L^{+0}(v_i;P)$.*
(ii) *For a vertex $v_i \in P$ such that $v_i \leq x_{\rm opt}(P)$, $\Theta_R^{-0}(v_i;P) \geq \Theta(v_i) \geq \Theta_R(v_i;P)$.*

Proof: Here, we prove only (i) ((ii) can be similarly proved). Let us look at a vertex $v_i \in P$ such that $v_i \geq x_{\rm opt}(P)$ (see Fig. 3(b)). By definition of $\Theta(v_i)$, we have $\Theta(v_i) \geq \Theta_L(v_i;P)$. Thus, in order to prove (i), we only need to show that

$$\Theta(v_i) \leq \Theta_L^{+0}(v_i;P). \tag{7}$$

By the condition of $v_i \geq x_{\rm opt}(P)$, $\Theta_L^{+0}(v_i;P) \geq \Theta_R(v_i;P)$ holds. Therefore, if $\Theta(v_i) = \Theta_R(v_i;P)$, (7) holds. If $\Theta(v_i) = \Theta_L(v_i;P)$, (7) also holds by (3) and (5). Otherwise, for a sink location $x = v_i$, an evacuee who lastly reaches v_i arrives at v_i through some adjacent vertex $u \in \delta(v_i)$ which is not on P. Suppose that we move the sink location from $x = v_i$ towards a point along P with distance ϵ in the right direction (i.e., $x = v_i + \epsilon$) where ϵ is a sufficiently small positive number. Then, the last evacuee first reaches v_i at time $\Theta(v_i)$, may be blocked there, and eventually reaches $x = v_i + \epsilon$, thus, he/she can reach $x = v_i + \epsilon$ after time $\Theta(v_i) + \epsilon\tau$, that is, $\Theta(v_i) + \epsilon\tau \leq \Theta_L(v_i + \epsilon; P)$ holds. By definition of (5), we obtain (7). □

Proof of Lemma 1: By Claims 1 and 2, $\Theta(x)$ is always unimodal in x along P although it may be discontinuous at v_i for $i \in [1,k-1]$. □

Proof of Lemma 2: Let us consider a path P from a leaf to another leaf through adjacent vertices v and $\hat{u}$ where $\hat{u} = \text{argmax}\{\Theta(v,u) \mid u \in \delta(v)\}$. Let us define the left direction in P as the direction from v to $\hat{u}$ and the right direction as the other one. Suppose that there are $k+1$ vertices $v_0, v_1, \ldots, v_k$ in P, and $v = v_i$ and $\hat{u} = v_{i-1}$ with $i \in [1, k-1]$. We consider a point $p \in P$ such that $p = v_i + \epsilon$ with sufficiently small $\epsilon > 0$. If we can show $\Theta(v_i) < \Theta(p)$, there never exists x_{opt} in the right direction from v_i along P by Lemma 1. Then, this lemma can be proved by repeatedly applying the same discussion to all the other paths through v and $\hat{u}$. By the assumption of $\Theta(v_i) = \Theta_L(v_i; P)$, $\Theta_L(v_i; P) \geq \Theta_R(v_i; P)$ holds, and by (3), $\Theta_L(v_i; P) + \epsilon\tau \leq \Theta_L(p; P)$ and $\Theta_R(v_i; P) = \Theta_R(p; P) + \epsilon\tau$, that is, $\Theta_L(v_i; P) < \Theta_L(p; P)$ and $\Theta_R(v_i; P) > \Theta_R(p; P)$ hold. Thus, we have $\Theta_R(p; P) < \Theta_L(p; P)$, which implies that

$$\Theta(p) = \Theta_L(p; P). \tag{8}$$

From (8) and the above mentioned two facts $\Theta(v_i) = \Theta_L(v_i; P)$ and $\Theta_L(v_i; P) < \Theta_L(p; P)$, we derive $\Theta(v_i) < \Theta(p)$. □

2.3 Algorithm

In this section, we present an $O(n \log n)$ time algorithm for the minimum cost sink location problem in dynamic tree networks with uniform capacity, which we call BST (Binary Search in Tree).

First, we introduce the concept of *centroid of a tree* [12].

Definition 1 *For an undirected tree $T = (V, E)$, a centroid of T is a vertex which minimizes* $\max\{|V(v,u)| \mid u \in \delta(v)\}$ *for all $v \in V$.*

Kang et al. [12] showed that a centroid m of T can be computed in $O(|V|)$ time and

$$\max\{|V(m,u)| \mid u \in \delta(m)\} \leq \frac{|V|}{2} \tag{9}$$

holds.

Let us explain at the first iteration by algorithm BST. Letting $U_1 = V$, the algorithm first finds a centroid m_1 of $T(U_1)$ and computes $d(m_1, v)$ for every $v \in U_1$. Then, in order to compute $\Theta(m_1, u)$ for each $u \in \delta(m_1)$, the algorithm basically creates the list $L(u)$ of all vertices $v \in U_1 \cap V(m_1, u)$ which are arranged in the nondecreasing order of $d(m_1, v)$. From (3), we can derive that $\Theta(m_1, u)$ can be computed by using $L(u)$. In this manner, the algorithm computes $u_1 = \text{argmax}\{\Theta(m_1, u) \mid u \in \delta(m_1)\}$. After that, it sets $V_1 = U_1 \setminus (V(m_1, u_1) \cup \{m_1\})$ and merges lists $L(u)$ for $u \in \delta(m_1) \setminus \{u_1\}$ into a new list L_1. At the end of the first iteration, the algorithm sets $U_2 = U_1 \cap (V(m_1, u_1) \cup \{m_1\})$. Note that by Lemma 2, there exists x_{opt} in $T(U_2)$ and by (9), $|U_2| \leq |U_1|/2 + 1$ holds.

The algorithm iteratively performs the same procedure (see Fig. 4). More precisely, at the i-th iteration, it finds a centroid m_i of $T(U_i)$, computes $u_i =$

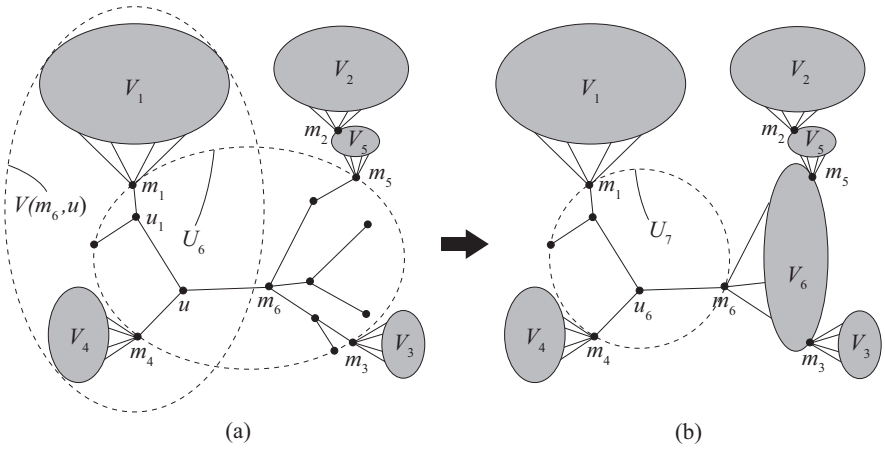

Figure 4: Illustration of the i-th iteration: **(a)** $i = 6$ and **(b)** $i = 7$

argmax$\{\Theta(m_i, u) \mid u \in \delta(m_i)\}$, sets $V_i = U_i \setminus (V(m_i, u_i) \cup \{m_i\})$, creates a list L_i of vertices $v \in V_i$ arranged in the nondecreasing order of $d(m_i, v)$ and also sets $U_{i+1} = U_i \cap (V(m_i, u_i) \cup \{m_i\})$. Since, at each iteration, the algorithm reduces the subgraph where $x_{\rm opt}$ exists so that the size becomes half or less roughly, it halts after $l = O(\log |V|)$ iterations. At this point, it finds two vertices m_l and $u_l \in U_l$ connected by an edge on which $x_{\rm opt}$ lies. Then, $x_{\rm opt}$ can be computed as follows. Let $x(t)$ denote a point dividing the edge (m_l, u_l) with the ratio of t to $1-t$ for some t $(0 \le t \le 1)$, and $\Theta(x(t), m_l)$ (resp. $\Theta(x(t), u_l)$) denote the minimum time required for all evacuees passing through m_l (resp. u_l) to complete the evacuation to $x(t)$. Then, $\Theta(x(t), m_l)$ and $\Theta(x(t), u_l)$ can be represented as follows:

$$\Theta(x(t), m_l) = \Theta(u_l, m_l) - (1-t)d(m_l, u_l)\tau, \quad (10)$$

$$\Theta(x(t), u_l) = \Theta(m_l, u_l) - td(m_l, u_l)\tau. \quad (11)$$

If there exists t such that $\Theta(x(t), m_l) = \Theta(x(t), u_l)$ and $0 \le t \le 1$, $x_{\rm opt} = x(t)$ holds by the unimodality of $\Theta(x)$. If the solution of $\Theta(x(t), m_l) = \Theta(x(t), u_l)$ satisfies $t < 0$, then $\Theta(m_l, u_l) < \Theta(u_l, m_l) - d(m_l, u_l)\tau$ holds, which implies $x_{\rm opt} = m_l$. Similarly, if $t > 1$, $x_{\rm opt} = u_l$ holds. Therefore, the algorithm can correctly output the optimal sink location $x_{\rm opt}$.

Now, let us analyze the time complexity of algorithm BST. We first show that the running time is $O(n \log^2 n)$ which will be improved to $O(n \log n)$ later, where $n = |V|$. Let us examine the running time for each iteration required by the algorithm. At the i-th iteration for $i \ge 2$, a centroid m_i of $T(U_i)$ can be found in $O(|U_i|)$ time (in [12]), and $d(m_i, v)$ can be computed for all $v \in V$ by depth-first search in $O(n)$ time. In the following, we consider two lists of vertices in $V(m_i, u)$ for $u \in \delta(m_i)$ which are arranged in the nondecreasing order of the distance from m_i, that is, $L(u)$ and $L'(u)$. Only one difference between $L(u)$ and $L'(u)$ is that $L(u)$ just consists of vertices in $U_i \cap V(m_i, u)$

although $L'(u)$ consists of all vertices in $V(m_i, u)$. If the algorithm creates a list $L'(u)$, $\Theta(m_i, u)$ can be computed as mentioned above. Each list $L'(u)$ can be created by a simple merge sort in $O(|V(m_i, u)| \log |V(m_i, u)|)$ time, so $u_i = \text{argmax}\{\Theta(m_i, u) \mid u \in \delta(m_i)\}$ can be computed in $O(n \log n + n)$ time. Therefore, in each iteration, it takes $O(|U_i| + n + n \log n + n) = O(n \log n)$ time. Since the algorithm halts after $O(\log n)$ iterations as mentioned above, our problem can be solved in $O(n \log^2 n)$ time.

Next, we show that the running time required to create lists $L'(u)$ for $u \in \delta(m_i)$ can be improved from $O(n \log n)$ to $O(n + |U_i| \log |U_i|)$. We first show the following claim.

Claim 3 *$|U_i| = O(\frac{n}{2^{i-1}})$ and $|V_i| = O(\frac{n}{2^{i-1}})$ hold for $i \geq 1$.*

Proof: By definition of U_i, we can clearly see that $|U_i| = O(n/2^{i-1})$ holds. Remind that $V_i = U_i \setminus (V(m_i, u_i) \cup \{m_i\})$ and $|U_i \cap V(m_i, u_i)| = O(|U_i|/2)$, thus we have $|V_i| = O(n/2^{i-1})$. □

The idea to improve the running time is to use the sorted lists L_j with $j = 1, 2, \ldots, i-1$. Let us look at Fig. 4(a), and focus on a vertex $u \in \delta(m_6)$ in the figure. The computation of $L'(u)$ can be done in $O(n \log n)$ time if we know $d(m_6, v)$ for all $v \in V(m_6, u)$. But, since $V(m_6, u) = V_1 \cup V_4 \cup (U_6 \cap V(m_6, u))$ holds and we have already computed L_1 and L_4, $L'(u)$ can be obtained faster if we only create a list $L(u)$ by computing $d(m_6, v)$ for all $v \in U_6 \cap V(m_6, u)$. Note that by (9), $|U_6 \cap V(m_6, u)|$ is at most $|U_6|/2$, which is about $|V_1|/64$ or $|V_4|/8$ by Claim 3, so its size is much smaller than $|V(m_6, u)|$. The idea is formalized as follows. For each $u \in \delta(m_i)$, the algorithm first creates a list $L(u)$ of vertices in $U_i \cap V(m_i, u)$, which takes $O(n' \log n')$ time where $n' = |U_i \cap V(m_i, u)|$. Thus, lists $L(u)$ for all $u \in \delta(m_i)$ can be created in $O(|U_i| \log |U_i|)$ time. For each $u \in \delta(m_i)$, the algorithm merges $L(u)$ and all lists L_j with $V_j \subseteq V(m_i, u)$ into $L'(u)$ (at this point, all of the original lists are maintained since these will be used later). For this merging operation, if we apply a simple merge sort, it takes $O(|V(m_i, u)| \log |V(m_i, u)|)$ time, which does not improve the running time. Here, we notice that $|L_j| = |V_j|$ for $j \in [1, i-1]$. Instead, the algorithm basically takes the following two steps to create each list $L'(u)$ for $u \in \delta(m_i)$:
[Step 1] For L_j such that $V_j \subseteq V(m_i, u)$, choose $L_p = \text{argmin}\{|L_j| \mid V_j \subseteq V(m_i, u)\}$ and merge each L_j in the increasing order of size (i.e., the decreasing order of j) with L_p one by one.
[Step 2] Merge the list obtained at Step 1 and $L(u)$ into $L'(u)$.
For all $u \in \delta(m_i)$, Step 1 takes in $O(\sum_{j=1}^{i-1} jn/2^{j-1}) = O(n)$ time, and thus, Step 2 takes $O(n + |U_i|) = O(n)$ time. Recall that $L(u)$ for all $u \in \delta(m_i)$ can be created in $O(|U_i| \log |U_i|)$ time. Then, by Claim 3, it takes $O(n + |U_i| \log |U_i|) = O(n + (n/2^{i-1}) \log(n/2^{i-1}))$ time to create lists $L'(u)$ for all $u \in \delta(m_i)$.

Lemma 3 *The i-th iteration of algorithm* BST *takes $O(n + \frac{n}{2^{i-1}} \log \frac{n}{2^{i-1}})$ time.*

Recall that the algorithm halts after $O(\log n)$ iterations. Thus, by Lemma 3, it takes $O(n \log n + \sum\{(n/2^{i-1}) \log(n/2^{i-1}) \mid i \in [1, \log n]\}) = O(n \log n)$ time for the entire iterations. Therefore, we obtain the following theorem.

Theorem 1 *The minimum cost sink location problem in a dynamic tree network with uniform capacity can be solved in $O(n \log n)$ time.*

3 Minimax regret sink location problem in dynamic tree networks with uniform capacity

Let $\mathcal{N} = (T, l, W, c, \tau)$ be a dynamic tree network with the underlying graph being a tree $T = (V, E)$, where l, c and τ are functions which are the same as those defined in Section 2, and W is also a function that associates each vertex $v \in V$ with an interval of integral supply denoted by $W(v) = [w^-(v), w^+(v)]$ with $0 < w^-(v) \leq w^+(v)$. Let $\mathcal{S}$ denote the Cartesian product of all $W(v)$ for $v \in V$ (i.e., a set of scenarios):

$$\mathcal{S} = \prod_{v \in V} [w^-(v), w^+(v)]. \tag{12}$$

When a scenario $s \in \mathcal{S}$ is given, we use the notation $w^s(v)$ to denote the supply of a vertex $v \in V$ under the scenario s.

For a sink location x given at a point in T and a given scenario $s \in \mathcal{S}$, let $\Theta^s(x)$ denote the minimum time required for all evacuees on T to complete the evacuation to x under s. For $u \in \delta(x)$, let $\Theta^s(x, u)$ denote the minimum time required for all evacuees on $T(x, u)$ to complete the evacuation to x. Then, we have

$$\Theta^s(x) \quad = \quad \max\{\Theta^s(x, u) \mid u \in \delta(x)\}. \tag{13}$$

For $\hat{u} = \mathrm{argmax}\{\Theta^s(x, u) \mid u \in \delta(x)\}$, we also have by (3)

$$\Theta^s(x, \hat{u}) = \max_{j \in [1, n']} \left\{ d(x, v_j)\tau + \left\lceil \frac{\sum_{i \in [j, n']} w^s(v_i)}{c} \right\rceil - 1 \right\}, \tag{14}$$

where n' is the number of vertices in $T(x, \hat{u})$ and $v_1 (= \hat{u}), v_2, \ldots, v_{n'}$ are vertices in $T(x, \hat{u})$ such that $d(x, v_i) \leq d(x, v_{i+1})$ for $1 \leq i \leq n' - 1$. In the following discussion, we assume $c = 1$ and omit the constant term (i.e., -1) from the above equations for the ease of exposition. Then, we have

$$\Theta^s(x, \hat{u}) = \max_{j \in [1, n']} \left\{ d(x, v_j)\tau + \sum_{i \in [j, n']} w^s(v_i) \right\}, \tag{15}$$

Here, let x^s_{opt} denote a point in T which minimizes $\Theta^s(x)$ under a scenario $s \in \mathcal{S}$. In the following, we use the notation Θ^s_{opt} for a scenario $s \in \mathcal{S}$ to denote $\Theta^s(x^s_{\mathrm{opt}})$. We now define the *regret* for x under s as

$$R^s(x) \quad = \quad \Theta^s(x) - \Theta^s_{\mathrm{opt}}. \tag{16}$$

Moreover, we also define the *maximum regret* for x as

$$R_{\max}(x) \quad = \quad \max\{R^s(x) \mid s \in \mathcal{S}\}. \tag{17}$$

If $\hat{s} = \text{argmax}\{R^s(x) \mid s \in \mathcal{S}\}$, we call $\hat{s}$ the *worst case scenario* for a sink location x. The goal is to find a point $x^* \in T$, called the *minimax regret sink location*, which minimizes $R_{\max}(x)$ over $x \in T$, i.e., the objective is to

$$\text{minimize } \{R_{\max}(x) \mid x \in T\}. \tag{18}$$

3.1 Properties

First, we define a set of so-called *dominant scenarios* for a vertex $v \in V$ among which the worst case scenario exists when the sink is located at v. Suppose that u is a vertex adjacent to v, n' is the number of vertices in $T(v,u)$ and $v_1(=u), v_2, \ldots, v_{n'}$ are vertices in $T(v,u)$ such that $d(v,v_i) \leq d(v,v_{i+1})$ for $1 \leq i \leq n'-1$. We now consider a scenario $s \in \mathcal{S}$ such that $w^s(v_i) = w^+(v_i)$ for $v_i \in T(v,u)$ such that $l \leq i \leq n'$ with some $l \in [1,n']$ and $w^s(v') = w^-(v')$ for all the other vertices $v' \in V$. In the following, such a scenario is said to be *dominant* for v, and represented by $s(v,v_l)$. Then, let $\mathcal{S}_{\mathrm{d}}(v,u) = \{s(v,v_l) \mid l \in [1,n']\}$, and also let $\mathcal{S}_{\mathrm{d}}(v) = \bigcup_{u \in \delta(v)} \mathcal{S}_{\mathrm{d}}(v,u)$. Note that $\mathcal{S}_{\mathrm{d}}(v)$ consists of $n-1$ scenarios. The following is a key lemma, which can be obtained from a lemma proved in [6, 10].

Lemma 4 *If a sink is located at a vertex $v \in V$, there exists a worst case scenario for v which belongs to $\mathcal{S}_{\mathrm{d}}(v)$.*

Proof: Suppose that $\hat{s} = \text{argmax}\{R^s(v) \mid s \in \mathcal{S}\}$, $\hat{u} = \text{argmax}\{\Theta^{\hat{s}}(v,u) \mid u \in \delta(v)\}$, n' is the number of vertices in $T(v,\hat{u})$, $v_1(=\hat{u}), v_2, \ldots, v_{n'}$ are vertices in $T(v,\hat{u})$ such that $d(v,v_i) \leq d(v,v_{i+1})$ for $1 \leq i \leq n'-1$ and

$$l = \underset{j \in [1,n']}{\text{argmax}} \left\{ d(v,v_j)\tau + \sum_{i \in [j,n']} w^{\hat{s}}(v_i) \right\}, \tag{19}$$

that is,

$$\Theta^{\hat{s}}(v,\hat{u}) = d(v,v_l)\tau + \sum_{i \in [l,n']} w^{\hat{s}}(v_i). \tag{20}$$

Here, let us consider a dominant scenario $s(v,v_l)$. Then, we prove that $R^{s(v,v_l)}(v) \geq R^{\hat{s}}(v)$ holds. If $\hat{s}$ is not equal to $s(v,v_l)$, we have two cases, i.e.,

- **(I)** there exists a vertex $v' \in V(v,\hat{u})$ such that $d(v,v_l) \leq d(v,v') \leq d(v,v_{n'})$ and $w^{\hat{s}}(v') < w^+(v')$,
- **(II)** there exists a vertex $v' \in V(v,\hat{u})$ such that $d(v,v_1) \leq d(v,v') < d(v,v_l)$ and $w^{\hat{s}}(v') > w^-(v')$ or $v' \in V \setminus V(v,\hat{u})$ such that $w^{\hat{s}}(v') > w^-(v')$.

For (I), we consider another scenario $\hat{s}_+$ such that $w^{\hat{s}_+}(v') = w^+(v')$ and $w^{\hat{s}_+}(v) = w^{\hat{s}}(v)$ for $v \in V \setminus \{v'\}$. For (II), we similarly consider $\hat{s}_-$ such that $w^{\hat{s}_-}(v') = w^-(v')$ and $w^{\hat{s}_-}(v) = w^{\hat{s}}(v)$ for $v \in V \setminus \{v'\}$. If we can show that

$R^{\hat{s}_+}(v) \geq R^{\hat{s}}(v)$ holds for (I) and $R^{\hat{s}_-}(v) \geq R^{\hat{s}}(v)$ holds for (II), we will eventually obtain $R^{s(v,v_l)}(v) \geq R^{\hat{s}}(v)$ by repeatedly applying the same discussion as long as there exists such a vertex v'.
(I): Let $\Delta = w^+(v') - w^{\hat{s}}(v')$. We first notice $\Theta^{\hat{s}_+}(v) = \Theta^{\hat{s}_+}(v, \hat{u})$ and $\Theta^{\hat{s}_+}(v, \hat{u})$ $= d(v, v_l)\tau + \sum_{i \in [l,n']} w^{\hat{s}_+}(v_i) = \Theta^{\hat{s}}(v, \hat{u}) + \Delta$ by (15) and (20). Thus, we have

$$\Theta^{\hat{s}_+}(v) = \Theta^{\hat{s}}(v) + \Delta. \tag{21}$$

By the optimality of $x^{\hat{s}_+}_{\rm opt}$ under $\hat{s}_+$, $\Theta^{\hat{s}_+}_{\rm opt} \leq \Theta^{\hat{s}_+}(x^{\hat{s}}_{\rm opt})$ holds. Here, we claim that $\Theta^{\hat{s}_+}(p) \leq \Theta^{\hat{s}}(p) + \Delta$ holds for any point $p \in T$, so $\Theta^{\hat{s}_+}(x^{\hat{s}}_{\rm opt}) \leq \Theta^{\hat{s}}_{\rm opt} + \Delta$ holds. Thus, we have

$$\Theta^{\hat{s}_+}_{\rm opt} \leq \Theta^{\hat{s}}_{\rm opt} + \Delta. \tag{22}$$

By (16), (21) and (22), we obtain $R^{\hat{s}_+}(v) \geq R^{\hat{s}}(v)$.
(II): In this case, $\Theta^{\hat{s}_-}(v) = \Theta^{\hat{s}_-}(v, \hat{u})$ and $\Theta^{\hat{s}_-}(v, \hat{u}) = \Theta^{\hat{s}}(v, \hat{u})$ by (15) and (20). Thus, we have

$$\Theta^{\hat{s}_-}(v) = \Theta^{\hat{s}}(v). \tag{23}$$

By the optimality of $x^{\hat{s}_-}_{\rm opt}$ under $\hat{s}_-$, $\Theta^{\hat{s}_-}_{\rm opt} \leq \Theta^{\hat{s}_-}(x^{\hat{s}}_{\rm opt})$ holds. Here, we claim that $\Theta^{\hat{s}_-}(x^{\hat{s}}_{\rm opt}) \leq \Theta^{\hat{s}}_{\rm opt}$ holds, we thus have

$$\Theta^{\hat{s}_-}_{\rm opt} \leq \Theta^{\hat{s}}_{\rm opt}. \tag{24}$$

By (16), (23) and (24), we obtain $R^{\hat{s}_-}(v) \geq R^{\hat{s}}(v)$. □

Here, we have the following claim by Lemma 1.

Claim 4 *For a scenario $s \in \mathcal{S}$, function $\Theta^s(x)$ is unimodal in x when x moves along a path from a leaf to another leaf in T.*

For a given scenario $s \in \mathcal{S}$, by the definition of (16) and Claim 4, function $R^s(x)$ is unimodal in x along a path from a leaf to another leaf in T. Thus, function $R_{\max}(x)$ is also unimodal in x since it is the upper envelope of unimodal functions by (17).

Lemma 5 *Along a path from a leaf to another leaf in T, function $R_{\max}(x)$ is unimodal in x.*

We also have the following claim by Lemma 2.

Claim 5 *For a scenario $s \in \mathcal{S}$ and a vertex $v \in V$, if $\hat{u} = \text{argmax}\ \{\Theta^s(v, u) \mid u \in \delta(v)\}$ holds, there exists $x^s_{\rm opt} \in T(V(v, \hat{u}) \cup \{v\})$.*

Here, suppose that $\hat{s} = \text{argmax}\{R^s(v) \mid s \in \mathcal{S}\}$ and $\hat{u} = \text{argmax}\{\Theta^{\hat{s}}(v, u) \mid u \in \delta(v)\}$ hold for a vertex $v \in V$. We now show that there also exists the minimax regret sink location x^* in $T(V(v, \hat{u}) \cup \{v\})$. Suppose otherwise: there exists x^* in $T(v, u)$ or on an edge (v, u) (not including endpoints) for some $u \in \delta(v)$ with

$u \neq \hat{u}$. By Claim 5, there exists $x^{\hat{s}}_{\mathrm{opt}}$ in $T(V(v,\hat{u}) \cup \{v\})$. Now, let us consider a path which goes through $x^{\hat{s}}_{\mathrm{opt}}$, v and x^* in this order. Then, by Claim 4, $\Theta^{\hat{s}}(x)$ is increasing in x when x moves along this path from $x^{\hat{s}}_{\mathrm{opt}}$ to x^*, which implies that $\Theta^{\hat{s}}(x^*) > \Theta^{\hat{s}}(v)$ holds. Thus, $R^{\hat{s}}(x^*) > R^{\hat{s}}(v)$ also holds by (16). We have $R_{\max}(x^*) \geq R^{\hat{s}}(x^*)$ by the maximality of $R_{\max}(x^*)$ and $R^{\hat{s}}(v) = R_{\max}(v)$ by the definition of $\hat{s}$, thus $R_{\max}(x^*) > R_{\max}(v)$ holds, which contradicts the optimality of x^*. By the above discussion, we obtain the following lemma.

Lemma 6 *For a vertex $v \in V$, if $\hat{s} = \mathrm{argmax}\{R^s(v) \mid s \in \mathcal{S}\}$ and $\hat{u} = \mathrm{argmax}\{\Theta^{\hat{s}}(v,u) \mid u \in \delta(v)\}$ hold, there exists the minimax regret sink location $x^* \in T(V(v,\hat{u}) \cup \{v\})$.*

3.2 Algorithm

In this section, we present an $O(n^2 \log^2 n)$ time algorithm that computes $x^* \in T$ which minimizes function $R_{\max}(x)$.

We first show how to compute $R_{\max}(v)$ for a given vertex $v \in V$. Given a dominant scenario $s \in \mathcal{S}_{\mathrm{d}}(v)$, $\Theta^s(v)$ can be computed in $O(n \log n)$ time, and by Theorem 1, Θ^s_{opt} can be computed in $O(n \log n)$ time. Thus by (16), $R^s(v)$ can be computed in $O(n \log n)$ time. By Lemma 4, we only need to consider $n-1$ dominant scenarios for v, thus, $R_{\max}(v)$ can be computed by (17) in $O(n^2 \log n)$ time.

Lemma 7 *For a vertex $v \in V$, $R_{\max}(v)$ can be computed in $O(n^2 \log n)$ time.*

In order to find the minimax regret sink location $x^* \in T$, we apply an algorithm similar to the one presented at Section 2.3. The algorithm maintains a vertex set $U_i \subseteq V$ which induces a connected subgraph of T including x^*. At the beginning of the procedure, the algorithm sets $U_1 = V$, and at i-th iteration, it finds a centroid m_i of $T(U_i)$, computes $R_{\max}(m_i)$ in the above mentioned manner, and sets $U_{i+1} = U_i \cap (V(m_i,u_i) \cup \{v\})$ where $u_i = \mathrm{argmax}\{\Theta^{\hat{s}}(m_i,u) \mid u \in \delta(m_i)\}$ and $\hat{s} = \mathrm{argmax}\{R^s(m_i) \mid s \in \mathcal{S}_{\mathrm{d}}(m_i)\}$. Note that, by Lemma 6, $T(U_{i+1})$ contains x^* if $T(U_i)$ includes x^*. The algorithm iteratively performs the same procedure until $|U_l|$ becomes two where $l = O(\log n)$. Suppose that there eventually remain two vertices m_l and $u_l \in U_l$. Then, the algorithm has already known that

$$R_{\max}(m_l) = \Theta^{\hat{s}_1}(m_l) - \Theta^{\hat{s}_1}_{\mathrm{opt}}, \tag{25}$$

$$R_{\max}(u_l) = \Theta^{\hat{s}_2}(u_l) - \Theta^{\hat{s}_2}_{\mathrm{opt}}, \tag{26}$$

$$m_l = \mathrm{argmax}\{\Theta^{\hat{s}_2}(u_l,u) \mid u \in \delta(u_l)\}, \tag{27}$$

$$u_l = \mathrm{argmax}\{\Theta^{\hat{s}_1}(m_l,u) \mid u \in \delta(m_l)\}, \tag{28}$$

where $\hat{s}_1$ and $\hat{s}_2$ are worst case scenarios for m_l and u_l, respectively. Let $x(t)$ denote a point dividing the edge (m_l,u_l) with the ratio of t to $1-t$ for some t $(0 \leq t \leq 1)$. If there exists t such that $R_{\max}(m_l) - td(m_l,u_l)\tau = R_{\max}(u_l) - (1-t)d(m_l,u_l)\tau$ and $0 \leq t \leq 1$, $x^* = x(t)$ holds by the unimodality of $R_{\max}(x)$.

If the solution satisfies $t < 0$, then $R_{\max}(m_l) < R_{\max}(u_l) - d(m_l, u_l)\tau$ holds, which implies $x^* = m_l$. Similarly, if $t > 1$, $x^* = u_l$ holds. As above, the algorithm correctly outputs the minimax regret sink location x^* after $O(\log n)$ iterations. Thus, by Lemmas 6 and 7, we obtain the following theorem.

Theorem 2 *The minimax regret sink location problem in a dynamic tree network with uniform capacity can be solved in $O(n^2 \log^2 n)$ time.*

4 Conclusion

In this paper, we developed an $O(n^2 \log^2 n)$ time algorithm for the minimax regret sink location problem in dynamic tree networks with uniform capacity. We also developed an $O(n \log n)$ time algorithm for the minimum cost sink location problem in dynamic tree networks with uniform capacity.

Here, recall that each input value is given as an integer and each function always returns an integer in this paper, which is called *the discrete model*. On the other hand, if each input value is given as a real number and ceiling is removed from each function, the model is called *the continuous model* [14]. Indeed we have solved the problem in the discrete model with $c = 1$, the case of $c \geq 2$ is still left open. Recently, it turns out that in the discrete model with $c \geq 2$, there may exist a sink location for which any worst case scenario is not dominant, which implies that the algorithm has to consider more than $O(n)$ scenarios for a fixed sink (Guru Prakash and Prashanth Srikanthan and the authors of this paper, private communication, 2014). However, in the continuous model, all claims, lemmas and the main theorem in Section 3 still hold even if $c \geq 2$, therefore we can directly apply the proposed algorithm to the continuous model without increasing the time complexity. Also, we can prove that the solution for the continuous model can be regarded as an approximation for the discrete model such that the difference between the approximate cost and the optimal cost is at most 1 (details are omitted).

In addition, we leave as an open problem to extend the solvable networks for the minimax regret sink location problem to dynamic tree networks with general capacities. Indeed, under a fixed scenario, the algorithm by [14] can solve the minimum cost sink location problem in dynamic tree networks with general capacity, but we cannot simply apply this as a subroutine to solve the minimax regret sink location problem. For example, if Lemma 4 still holds in a dynamic tree network with general capacities, we can expect that an $O(n^2 \log^3 n)$ time algorithm will be achieved.

References

[1] I. Averbakh and O. Berman. Algorithms for the robust 1-center problem on a tree. *European Journal of Operational Research*, 123(2):292–302, 2000. doi:10.1016/S0377-2217(99)00257-X.

[2] B. K. Bhattacharya and T. Kameda. A linear time algorithm for computing minmax regret 1-median on a tree. In *Computing and Combinatorics - 18th Annual International Conference, COCOON 2012, Sydney, Australia, August 20-22, 2012. Proceedings*, pages 1–12, 2012. doi:10.1007/978-3-642-32241-9_1.

[3] B. K. Bhattacharya, T. Kameda, and Z. Song. A linear time algorithm for computing minmax regret 1-median on a tree network. *Algorithmica*, 70(1):2–21, 2014. doi:10.1007/s00453-013-9851-7.

[4] G. S. Brodal, L. Georgiadis, and I. Katriel. An o(nlogn) version of the averbakh-berman algorithm for the robust median of a tree. *Oper. Res. Lett.*, 36(1):14–18, 2008. doi:10.1016/j.orl.2007.02.012.

[5] B. Chen and C. Lin. Minmax-regret robust 1-median location on a tree. *Networks*, 31(2):93–103, 1998. doi:10.1002/(SICI)1097-0037(199803)31:2<93::AID-NET4>3.0.CO;2-E.

[6] S. Cheng, Y. Higashikawa, N. Katoh, G. Ni, B. Su, and Y. Xu. Minimax regret 1-sink location problems in dynamic path networks. In *Theory and Applications of Models of Computation, 10th International Conference, TAMC 2013, Hong Kong, China, May 20-22, 2013. Proceedings*, pages 121–132, 2013. doi:10.1007/978-3-642-38236-9_12.

[7] E. Conde. Minmax regret location-allocation problem on a network under uncertainty. *European Journal of Operational Research*, 179(3):1025–1039, 2007. doi:10.1016/j.ejor.2005.11.040.

[8] E. Conde. A note on the minmax regret centdian location on trees. *Oper. Res. Lett.*, 36(2):271–275, 2008. doi:10.1016/j.orl.2007.05.009.

[9] L. R. Ford Jr and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations research*, 6(3):419–433, 1958.

[10] Y. Higashikawa, J. Augustine, S.-W. Cheng, M. J. Golin, N. Katoh, G. Ni, B. Su, and Y. Xu. Minimax regret 1-sink location problem in dynamic path networks. *Theoretical Computer Science*, 2014.

[11] N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for evacuation problem in dynamic network flows with uniform arc capacity. *IEICE Transactions*, 89-D(8):2372–2379, 2006. doi:10.1093/ietisy/e89-d.8.2372.

[12] A. N. Kang and D. A. Ault. Some properties of a centroid of a free tree. *Information Processing Letters*, 4(1):18–20, 1975.

[13] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*, volume 14. Springer, 1997.

[14] S. Mamada, T. Uno, K. Makino, and S. Fujishige. An o(n $\log^2$n) algorithm for the optimal sink location problem in dynamic tree networks. *Discrete Applied Mathematics*, 154(16):2387–2401, 2006. doi:10.1016/j.dam.2006.04.010.

[15] W. Ogryczak. Conditional median as a robust solution concept for uncapacitated location problems. *Top*, 18(1):271–285, 2010.

[16] J. Puerto, A. M. Rodríguez-Chía, and A. Tamir. Minimax regret single-facility ordered median location problems on networks. *INFORMS Journal on Computing*, 21(1):77–87, 2009. doi:10.1287/ijoc.1080.0280.

[17] H. Wang. Minmax regret 1-facility location on uncertain path networks. *European Journal of Operational Research*, 239(3):636–643, 2014. doi:10.1016/j.ejor.2014.06.026.