Almost Linear Time Algorithms for Minsum k-Sink Problems on Dynamic Flow Path Networks

Yuya Higashikawa[1(✉)], Naoki Katoh[1], Junichi Teruyama[1], and Koji Watase[2]

[1] University of Hyogo, Kobe, Japan
{higashikawa,naoki.katoh,junichi.teruyama}@sis.u-hyogo.ac.jp
[2] Kwansei Gakuin University, Sanda, Japan
fnt43517@kwansei.ac.jp

Abstract. We address the facility location problems on dynamic flow path networks. A *dynamic flow path network* consists of an undirected path with positive edge lengths, positive edge capacities, and positive vertex weights. A path can be considered as a road, an edge length as the distance along the road and a vertex weight as the number of people at the site. An edge capacity limits the number of people that can enter the edge per unit time. In the dynamic flow network, given particular points on edges or vertices, called *sinks*, all the people evacuate from the vertices to the sinks as quickly as possible. The problem is to find the location of sinks on a dynamic flow path network in such a way that the aggregate evacuation time (i.e., the sum of evacuation times for all the people) to sinks is minimized. We consider two models of the problem: the *confluent flow model* and the *non-confluent flow model*. In the former model, the way of evacuation is restricted so that all the people at a vertex have to evacuate to the same sink, and in the latter model, there is no such restriction. In this paper, for both the models, we develop algorithms which run in almost linear time regardless of the number of sinks. It should be stressed that for the confluent flow model, our algorithm improves upon the previous result by Benkoczi et al. [Theoretical Computer Science, 2020], and one for the non-confluent flow model is the first polynomial time algorithm.

Keywords: Dynamic flow networks · Facility location problems · Minimum k-link path problem · Persistent data structures

1 Introduction

Recently, many disasters, such as earthquakes, nuclear plant accidents, volcanic eruptions and flooding, have struck in many parts of the world, and it has been recognized that orderly evacuation planning is urgently needed. A powerful tool

A full version of the paper is available at [14]; https://arxiv.org/abs/2010.05729.

W. Wu and Z. Zhang (Eds.): COCOA 2020, LNCS 12577, pp. 198–213, 2020.
https://doi.org/10.1007/978-3-030-64843-5_14

for evacuation planning is the *dynamic flow model* introduced by Ford and Fulkerson [10], which represents movement of commodities over time in a network. In this model, we are given a graph with *source* vertices and *sink* vertices. Each source vertex is associated with a positive weight, called a *supply*, each sink vertex is associated with a positive weight, called a *demand*, and each edge is associated with positive length and capacity. An edge capacity limits the amount of supply that can enter the edge per unit time. One variant of the dynamic flow problem is the *quickest transshipment problem*, of which the objective is to send exactly the right amount of supply out of sources into sinks with satisfying the demand constraints in the minimum overall time. Hoppe and Tardos [15] provided a polynomial time algorithm for this problem in the case where the transit times are integral. However, the complexity of their algorithm is very high. Finding a practical polynomial time solution to this problem is still open. A reader is referred to a recent survey by Skutella [18] on dynamic flows.

This paper discusses a related problem, called the *k-sink problem* [2,4,5,7–9,12,13,16], of which the objective is to find a location of k sinks in a given dynamic flow network so that all the supply is sent to the sinks as quickly as possible. For the optimality of location, the following two criteria can be naturally considered: the minimization of *evacuation completion time* and *aggregate evacuation time* (i.e., *average evacuation time*). We call the k-sink problem that requires finding a location of k sinks that minimizes the evacuation completion time (resp. the aggregate evacuation time) the *minmax* (resp. *minsum*) *k-sink problem*. Several papers have studied the minmax k-sink problems on dynamic flow networks [2,7–9,12,13,16]. On the other hand, the minsum k-sink problems on dynamic flow networks have not been studied except for the case of path networks [4,5,13].

Moreover, there are two models on the way of evacuation. Under the *confluent flow model*, all the supply leaving a vertex must evacuate to the same sink through the same edges, and under the *non-confluent flow model*, there is no such restriction. To our knowledge, all the papers which deal with the k-sink problems [2,4,5,7–9,13] adopt the confluent flow model.

In order to model the evacuation behavior of people, it might be natural to treat each supply as a discrete quantity as in [15,16]. Nevertheless, almost all the previous papers on sink problems [2,7–9,12,13] treat each supply as a continuous quantity since it is easier for mathematically handling the problems and the effect is small enough to ignore when the number of people is large. Throughout the paper, we also adopt the model with continuous supplies.

In this paper, we study the minsum k-sink problems on dynamic flow path networks under both the confluent flow model and the non-confluent flow model. A path network can model a coastal area surrounded by the sea and a hilly area, an airplane aisle, a hall way in a building, a street, a highway, etc., to name a few. For the confluent flow model, the previous best results are an $O(kn \log^3 n)$ time algorithm for the case with uniform edge capacity in [4], and $O(kn \log^4 n)$ time algorithm for the case with general edge capacities in [5], where n is the number of vertices on path networks. We develop algorithms which run in time $\min\{O(kn \log^2 n), n2^{O(\sqrt{\log k \log\log n})} \log^2 n\}$ for the case with uniform edge

capacity, and in time $\min\{O(kn \log^3 n), n2^{O(\sqrt{\log k \log\log n})} \log^3 n\}$ for the case with general edge capacities, respectively. Thus, our algorithms improve upon the complexities by [4,5] for any value of k. Especially, for the non-confluent flow model, this paper provides the first polynomial time algorithms.

Since the number of sinks k is at most n, we confirm $2^{O(\sqrt{\log k \log\log n})} = n^{O(\sqrt{\log\log n/\log n})} = n^{o(1)}$, which means that our algorithms are the first ones which run in almost linear time (i.e., $n^{1+o(1)}$ time) regardless of k. The reason why we could achieve almost linear time algorithms for the minsum k-sink problems is that we newly discover a convex property from a novel point of view. In all the previous papers on the k-sink problems, the evacuation completion. time and the aggregate evacuation time (called CT and AT, respectively) are basically determined as functions in "distance": Let us consider the case with a 1-sink. The values $\mathsf{CT}(x)$ or $\mathsf{AT}(x)$ may change as a sink location x moves along edges in the network. In contrast, we introduce a new metric for CT and AT as follows: assuming that a sink is fixed and all the supply in the network flows to the sink, for a positive real z, $\mathsf{CT}(z)$ is the time at which the first z of supply completes its evacuation to the sink and then $\mathsf{AT}(z)$ is the integral of $\mathsf{CT}(z)$, i.e., $\mathsf{AT}(z) = \int_0^z \mathsf{CT}(t)dt$. We can observe that $\mathsf{AT}(z)$ is convex in z since $\mathsf{CT}(z)$ is increasing in z. Based on the convexity of $\mathsf{AT}(z)$, we develop efficient algorithms.

The rest of the paper is organized as follows. In Sect. 2, we introduce the terms that are used throughout the paper and explain our models. In Sect. 3, we show that our problem can be reduced to the *minimum k-link path problem with links satisfying the concave Monge condition*. This immediately implies by Schieber [17] that the optimal solutions for our problems can be obtained by solving $\min\{O(kn), n2^{O(\sqrt{\log k \log\log n})}\}$ subproblems of computing the optimal aggregate evacuation time for subpaths, in each of which two sinks are located on its endpoints. Section 3 subsequently shows an overview of the algorithm that solves the above subproblems. In Sect. 4, we introduce novel data structures, which enable to solve each of the above subproblems in $O(\text{poly}\log n)$ time. Section 5 concludes the paper.

2 Preliminaries

2.1 Notations

For two real values a, b with $a < b$, let $[a, b] = \{t \in \mathbb{R} \mid a \leq t \leq b\}$, $[a, b) = \{t \in \mathbb{R} \mid a \leq t < b\}$, $(a, b] = \{t \in \mathbb{R} \mid a < t \leq b\}$, and $(a, b) = \{t \in \mathbb{R} \mid a < t < b\}$, where $\mathbb{R}$ is the set of real values. For two integers i, j with $i \leq j$, let $[i..j] = \{h \in \mathbb{Z} \mid i \leq h \leq j\}$, where $\mathbb{Z}$ is the set of integers. A dynamic flow path network $\mathcal{P}$ is given as a 5-tuple $(P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$, where P is a path with vertex set $V = \{v_i \mid i \in [1..n]\}$ and edge set $E = \{e_i = (v_i, v_{i+1}) \mid i \in [1..n-1]\}$, $\mathbf{w}$ is a vector $\langle w_1, \ldots, w_n \rangle$ of which a component w_i is the *weight* of vertex v_i representing the amount of supply (e.g., the number of evacuees, cars) located at v_i, $\mathbf{c}$ is a vector $\langle c_1, \ldots, c_{n-1} \rangle$ of which a component c_i is the *capacity* of edge e_i representing the upper bound on the flow rate through e_i per unit time, $\mathbf{l}$ is

a vector $\langle \ell_1, \ldots, \ell_{n-1} \rangle$ of which a component ℓ_i is the *length* of edge e_i, and τ is the time which unit supply takes to move unit distance on any edge.

We say a point p lies on path $P = (V, E)$, denoted by $p \in P$, if p lies on a vertex $v \in V$ or an edge $e \in E$. For two points $p, q \in P$, $p \prec q$ means that p lies to the left side of q. For two points $p, q \in P$, $p \preceq q$ means that $p \prec q$ or p and q lie on the same place. Let us consider two integers $i, j \in [1..n]$ with $i < j$. We denote by $P_{i,j}$ a *subpath* of P from v_i to v_j. Let $L_{i,j}$ be the distance between v_i and v_j, i.e., $L_{i,j} = \sum_{h=i}^{j-1} \ell_h$, and let $C_{i,j}$ be the minimum capacity for all the edges between v_i and v_j, i.e., $C_{i,j} = \min\{c_h \mid h \in [i..j-1]\}$. For $i \in [1..n]$, we denote the sum of weights from v_1 to v_i by $W_i = \sum_{j=1}^{i} w_j$. Note that, given a dynamic flow path network $\mathcal{P}$, if we construct two lists of W_i and $L_{1,i}$ for all $i \in [1..n]$ in $O(n)$ preprocessing time, we can obtain W_i for any $i \in [1..n]$ and $L_{i,j} = L_{1,j} - L_{1,i}$ for any $i, j \in [1..n]$ with $i < j$ in $O(1)$ time. In addition, $C_{i,j}$ for any $i, j \in [1..n]$ with $i < j$ can be obtained in $O(1)$ time with $O(n)$ preprocessing time, which is known as the *range minimum query* [1,3].

A *k-sink* $\mathbf{x}$ is k-tuple $(x_1, \ldots, x_k)$ of points on P, where $x_i \prec x_j$ for $i < j$. We define the function Id for point $p \in P$ as follows: the value $\mathrm{Id}(p)$ is an integer such that $v_{\mathrm{Id}(p)} \preceq p \prec v_{\mathrm{Id}(p)+1}$ holds. For a k-sink $\mathbf{x}$ for $\mathcal{P}$, a *divider* $\mathbf{d}$ is $(k-1)$-tuple $(d_1, \ldots, d_{k-1})$ of real values such that $d_i < d_j$ for $i < j$ and $W_{\mathrm{Id}(x_i)} \leq d_i \leq W_{\mathrm{Id}(x_{i+1})}$. Given a k-sink $\mathbf{x}$ and a divider $\mathbf{d}$ for $\mathcal{P}$, the portion $W_{\mathrm{Id}(x_i)} - d_{i-1}$ supply that originates from the left side of x_i flows to sink x_i, and the portion $d_i - W_{\mathrm{Id}(x_i)}$ supply that originates from the right side of x_i also flows to sink x_i. For instance, under the non-confluent flow model, if $W_{h-1} < d_i < W_h$ where $h \in [1..n]$, $d_i - W_{h-1}$ of w_h supply at v_h flows to sink x_i and the rest of $W_h - d_i$ supply to do sink x_{i+1}. The difference between the confluent flow model and the non-confluent flow model is that the confluent flow model requires that each value d_i of a divider $\mathbf{d}$ must take a value in $\{W_1, \ldots, W_n\}$, but the non-confluent flow model does not. For the notation, we set $d_0 = 0$ and $d_k = W_n$.

For a dynamic flow path network $\mathcal{P}$, a k-sink $\mathbf{x}$ and a divider $\mathbf{d}$, the *evacuation completion time* $\mathsf{CT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is the time at which all the supply completes the evacuation. The *aggregate evacuation time* $\mathsf{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is that the sum of the evacuation completion time for all the supply. Their explicit definitions are given later. In this paper, our task is, given a dynamic flow path network $\mathcal{P}$, to find a k-sink $\mathbf{x}$ and a divider $\mathbf{d}$ that minimize the aggregate evacuation time $\mathsf{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ in each evacuation model.

2.2 Aggregate Evacuation Time on a Path

For the confluent flow model, it is shown in [5,13] that for the minsum k-sink problems, there exists an optimal k-sink such that all the k sinks are at vertices. This fact also holds for the non-confluent flow model. Indeed, if a divider $\mathbf{d}$ is fixed, then we have k subproblems for a 1-sink and the optimal sink location for each subproblem is at a vertex. Thus, we have the following lemma.

Lemma 1 ([13]). *For the minsum k-sink problem in a dynamic flow path network, there exists an optimal k-sink such that all the k sinks are at vertices under the confluent/non-confluent flow model.*

Lemma 1 implies that it is enough to consider only the case that every sink is at a vertex. Thus, we suppose $\mathbf{x} = (x_1, \ldots, x_k) \in V^k$, where $x_i \prec x_j$ for $i < j$.

A Simple Example with a 1-sink. In order to give explicit definitions for the evacuation completion time and the aggregate evacuation time, let us consider a simple example for a 1-sink. We are given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices and set a unique sink on a vertex v_i, that is, $\mathbf{x} = (v_i)$ and $\mathbf{d} = ()$ which is the 0-tuple. In this case, all the supply on the left side of v_i (i.e., at $v_1, \ldots, v_{i-1}$) will flow right to sink v_i, and all the supply on the right side of v_i (i.e., at $v_{i+1}, \ldots, v_n$) will flow left to sink v_i. Note that in our models all the supply at v_i immediately completes its evacuation at time 0.

To deal with this case, we introduce some new notations. Let the function $\theta^{i,+}(z)$ denote the time at which the first $z - W_i$ of supply on the right side of v_i completes its evacuation to sink v_i (where $\theta^{i,+}(z) = 0$ for $z \in [0, W_i]$). Higashikawa [11] shows that the value $\theta^{i,+}(W_n)$, the evacuation completion time for all the supply on the right side of v_i, is given by the following formula:

$$\theta^{i,+}(W_n) = \max\left\{ \frac{W_n - W_{j-1}}{C_{i,j}} + \tau \cdot L_{i,j} \mid j \in [i+1..n] \right\}. \tag{1}$$

Recall that $C_{i,j} = \min\{c_h \mid h \in [i..j-1]\}$. We can generalize formula (1) to the case with any $z \in [0, W_n]$ as follows:

$$\theta^{i,+}(z) = \max\{\theta^{i,+,j}(z) \mid j \in [i+1..n]\}, \tag{2}$$

where $\theta^{i,+,j}(z)$ for $j \in [i+1..n]$ is defined as

$$\theta^{i,+,j}(z) = \begin{cases} 0 & \text{if } z \le W_{j-1}, \\ \frac{z - W_{j-1}}{C_{i,j}} + \tau \cdot L_{i,j} & \text{if } z > W_{j-1}. \end{cases} \tag{3}$$

Similarly, let $\theta^{i,-}(z)$ denote the time at which the first $W_{i-1} - z$ of supply on the left side of v_i completes its evacuation to sink v_i (where $\theta^{i,-}(z) = 0$ for $z \in [W_{i-1}, W_n]$). Then,

$$\theta^{i,-}(z) = \max\{\theta^{i,-,j}(z) \mid j \in [1..i-1]\}, \tag{4}$$

where $\theta^{i,-,j}(z)$ is defined as

$$\theta^{i,-,j}(z) = \begin{cases} \frac{W_j - z}{C_{j,i}} + \tau \cdot L_{j,i} & \text{if } z < W_j, \\ 0 & \text{if } z \ge W_j. \end{cases} \tag{5}$$

The aggregate evacuation times for the supply on the right side and the left side of v_i are

$$\int_{W_i}^{W_n} \theta^{i,+}(z)dz = \int_0^{W_n} \theta^{i,+}(z)dz \text{ and } \int_0^{W_{i-1}} \theta^{i,-}(z)dz = \int_0^{W_n} \theta^{i,-}(z)dz,$$

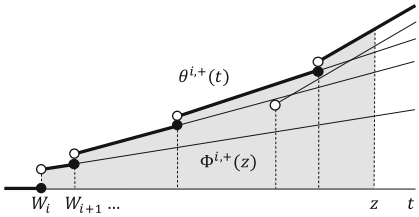

Fig. 1. The thick half-open segments indicate function $\theta^{i,+}(t)$ and the gray area indicates $\Phi^{i,+}(z)$ for some $z > W_i$.

respectively. Thus, the aggregate evacuation time $\mathsf{AT}(\mathcal{P}, (v_i), ())$ is given as

$$\mathsf{AT}(\mathcal{P}, (v_i), ()) = \int_0^{W_n} \left\{\theta^{i,+}(z) + \theta^{i,-}(z)\right\} dz.$$

Aggregate Evacuation Time with a k-Sink. Suppose that we are given a k-sink $\mathbf{x} = (x_1, \ldots, x_k) \in V^k$ and a divider $\mathbf{d} = (d_1, \ldots, d_{k-1})$. Recalling the definition of $\mathrm{Id}(p)$ for $p \in P$, we have $x_i = v_{\mathrm{Id}(x_i)}$ for all $i \in [1..k]$. In this situation, for each $i \in [1..k]$, the first $d_i - W_{\mathrm{Id}(x_i)}$ of supply on the right side of x_i and the first $W_{\mathrm{Id}(x_i)-1} - d_{i-1}$ of supply on the left side of x_i move to sink x_i.

By the argument of the previous section, the aggregate evacuation times for the supply on the right side and the left side of x_i are

$$\int_{W_{\mathrm{Id}(x_i)}}^{d_i} \theta^{\mathrm{Id}(x_i),+}(z)dz = \int_0^{d_i} \theta^{\mathrm{Id}(x_i),+}(z)dz \quad \text{and}$$
$$\int_{d_{i-1}}^{W_{\mathrm{Id}(x_i)-1}} \theta^{\mathrm{Id}(x_i),-}(z)dz = \int_{d_{i-1}}^{W_n} \theta^{\mathrm{Id}(x_i),-}(z)dz,$$

respectively. In order to give the general form for the above values, let us denote by $\Phi^{i,+}(z)$ the aggregate evacuation time when the first $z - W_i$ of supply on the right side of v_i flows to sink v_i. Similarly, we denote by $\Phi^{i,-}(z)$ the aggregate evacuation time when the first $W_{i-1} - z$ of supply on the left side of v_i flows to sink v_i. Therefore, we have

$$\Phi^{i,+}(z) = \int_0^z \theta^{i,+}(t)dt \quad \text{and} \quad \Phi^{i,-}(z) = \int_z^{W_n} \theta^{i,-}(t)dt = \int_{W_n}^z -\theta^{i,-}(t)dt \quad (6)$$

(see Fig. 1). Let us consider a subpath $P_{\mathrm{Id}(x_i),\mathrm{Id}(x_{i+1})}$ which is a subpath between sinks x_i and x_{i+1}. The aggregate evacuation time for the supply on $P_{\mathrm{Id}(x_i),\mathrm{Id}(x_{i+1})}$ is given by

$$\int_0^{d_i} \theta^{\mathrm{Id}(x_i),+}(z)dz + \int_{d_i}^{W_n} \theta^{\mathrm{Id}(x_{i+1}),-}(z)dz = \Phi^{\mathrm{Id}(x_i),+}(d_i) + \Phi^{\mathrm{Id}(x_{i+1}),-}(d_i).$$

For $i, j \in [1..n]$ with $i < j$, let us define

$$\Phi^{i,j}(z) = \Phi^{i,+}(z) + \Phi^{j,-}(z) = \int_0^z \theta^{i,+}(t)dt + \int_z^{W_n} \theta^{j,-}(t)dt \tag{7}$$

for $z \in [W_i, W_{j-1}]$. Then, the aggregate evacuation time $\mathsf{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is given as

$$\mathsf{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) = \Phi^{\mathrm{Id}(x_1),-}(0) + \sum_{i=1}^{k-1} \Phi^{\mathrm{Id}(x_i),\mathrm{Id}(x_{i+1})}(d_i) + \Phi^{\mathrm{Id}(x_k),+}(W_n). \tag{8}$$

In the rest of this section, we show the important properties of $\Phi^{i,j}(z)$. Let us first confirm that by Eq. (6), both $\Phi^{i,+}(z)$ and $\Phi^{j,-}(z)$ are convex in z since $\theta^{i,+}(z)$ and $-\theta^{j,-}(z)$ are non-decreasing in z, therefore $\Phi^{i,j}(z)$ is convex in z. On the condition of the minimizer for $\Phi^{i,j}(z)$, we have a more useful lemma.

Lemma 2. *For any $i, j \in [1..n]$ with $i < j$, there uniquely exists*

$$z^* \in \operatorname*{arg\,min}_{z \in [W_i, W_{j-1}]} \max\{\theta^{i,+}(z), \theta^{j,-}(z)\}.$$

Furthermore, $\Phi^{i,j}(z)$ is minimized on $[W_i, W_{j-1}]$ when $z = z^*$.

See Lemma 2 in [14] for the proof. In the following sections, such z^* is called the *pseudo-intersection point*[1] of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$, and we say that $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$ *pseudo-intersect* on $[W_i, W_{j-1}]$ at z^*.

3 Algorithms

In order to solve our problems, we reduce them to *minimum k-link path problems*. In the minimum k-link path problems, we are given a weighted complete directed acyclic graph (DAG) $G = (V', E', w')$ with $V' = \{v'_i \mid i \in [1..n]\}$ and $E' = \{(v'_i, v'_j) \mid i, j \in [1..n], i < j\}$. Each edge (v'_i, v'_j) is associated with weight $w'(i, j)$. We call a path in G a *k-link* path if the path contains exactly k edges. The task is to find a k-link path $(v'_{a_0} = v'_1, v'_{a_1}, v'_{a_2}, \ldots, v'_{a_{k-1}}, v'_{a_k} = v'_n)$ from v'_1 to v'_n that minimizes the sum of weights of k edges, $\sum_{i=1}^{k} w'(a_{i-1}, a_i)$. If the weight function w' satisfies the *concave Monge property*, then we can solve the minimum k-link path problems in almost linear time regardless of k.

Definition 1 (Concave Monge property). *We say function $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$ satisfies the concave Monge property if for any integers i, j with $i + 1 < j$, $f(i, j) + f(i + 1, j + 1) \leq f(i + 1, j) + f(i, j + 1)$ holds.*

Lemma 3 ([17]). *Given a weighted complete DAG with n vertices, if the weight function satisfies the concave Monge property, then there exists an algorithm that solves the minimum k-link path problem in time* $\min\{O(kn), n2^{O(\sqrt{\log k \log\log n})}\}$.

[1] The reason why we adopt a term "pseudo-intersection" is that two functions $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$ are not continuous in general while "intersection" is usually defined for continuous functions.

We describe how to reduce the k-sink problem on a dynamic flow path network $\mathcal{P} = (P = (V, E), \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices to the minimum $(k+1)$-link path problem on a weighted complete DAG $G = (V', E', w')$. We prepare a weighted complete DAG $G = (V', E', w')$ with $n+2$ vertices, where $V' = \{v'_i \mid i \in [0..n+1]\}$ and $E' = \{(v'_i, v'_j) \mid i, j \in [0..n+1], i < j\}$. We set the weight function w' as

$$w'(i,j) = \begin{cases} \mathsf{OPT}(i,j) & i,j \in [1..n], i < j, \\ \Phi^{i,+}(W_n) & i \in [1..n] \text{ and } j = n+1, \\ \Phi^{j,-}(0) & i = 0 \text{ and } j \in [1..n], \\ \infty & i = 0 \text{ and } j = n+1, \end{cases} \tag{9}$$

where $\mathsf{OPT}(i,j) = \min_{z \in [W_i, W_{j-1}]} \Phi^{i,j}(z)$.

Now, on a weighted complete DAG G made as above, let us consider a $(k+1)$-link path $(v'_{a_0} = v'_0, v'_{a_1}, \ldots, v'_{a_k}, v'_{a_{k+1}} = v'_{n+1})$ from v'_0 to v'_{n+1}, where $a_1, \ldots, a_k$ are integers satisfying $0 < a_1 < a_2 < \cdots < a_k < n+1$. The sum of weights of this $(k+1)$-link path is

$$\sum_{i=0}^{k} w'(a_i, a_{i+1}) = \Phi^{a_1,-}(0) + \sum_{i=1}^{k-1} \mathsf{OPT}(a_i, a_{i+1}) + \Phi^{a_k,+}(W_n).$$

This value is equivalent to $\min_{\mathbf{d}} \mathsf{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ for a k-sink $\mathbf{x} = (v_{a_1}, v_{a_2}, \ldots, v_{a_k})$ (recall Eq. (8)), which implies that a minimum $(k+1)$-link path on G corresponds to an optimal k-sink location for a dynamic flow path network $\mathcal{P}$.

We show in the following lemma that the function w' defined as formula (9) satisfies the concave Monge property under both of evacuation models. See Lemma 4 in [14] for the proof.

Lemma 4. *The weight function w' defined as formula (9) satisfies the concave Monge property under the confluent/non-confluent flow model.*

Lemmas 3 and 4 imply that if we can evaluate $w'(i,j)$ in time at most t for any $i, j \in [0..n+1]$ with $i < j$, then we can solve the k-sink problem in time $\min\{O(knt), n2^{O(\sqrt{\log k \log\log n})}t\}$.

In order to obtain $w'(i,j)$ for any $i, j \in [0..n+1]$ with $i < j$ in $O(\mathrm{poly}\log n)$ time, we introduce novel data structures and some modules using them. Basically, we construct a *segment tree* [6] $\mathcal{T}$ with root ρ such that its leaves correspond to indices of vertices of P arranged from left to right and its height is $O(\log n)$. For a node $u \in \mathcal{T}$, let $\mathcal{T}_u$ denote the subtree rooted at u, and let l_u (resp. r_u) denote the index of the vertex that corresponds to the leftmost (resp. rightmost) leaf of $\mathcal{T}_u$. Let p_u denote the parent of u if $u \neq \rho$. We say a node $u \in \mathcal{T}$ *spans* subpath P_{ℓ_u, r_u}. If $P_{\ell_u, r_u} \subseteq P'$ and $P_{\ell_{p_u}, r_{p_u}} \not\subseteq P'$, node u is called a *maximal subpath node* for P'. For each node $u \in \mathcal{T}$, let m_u be the number of edges in subpath P_{ℓ_u, r_u}, i.e., $m_u = r_u - \ell_u$. As with a standard segment tree, $\mathcal{T}$ has the following properties.

Property 1. For $i, j \in [1..n]$ with $i < j$, the number of maximal subpath nodes for $P_{i,j}$ is $O(\log n)$. Moreover, we can find all the maximal subpath nodes for $P_{i,j}$ by walking on $\mathcal{T}$ from leaf i to leaf j in $O(\log n)$ time.

Property 2. If one can construct data structures for each node u of a segment tree $\mathcal{T}$ in $O(f(m_u))$ time, where $f : \mathbb{N} \to \mathbb{R}$ is some function independent of n and bounded below by a linear function asymptotically, i.e., $f(m) = \Omega(m)$, then the running time for construction of data structures for every node in $\mathcal{T}$ is $O(f(n) \log n)$ time in total.

At each node $u \in \mathcal{T}$, we store four types of the information that depend on the indices of the vertices spanned by u, i.e., $l_u, \ldots, r_u$. We will introduce each type in Sect. 4. As will be shown there, the four types of the information at $u \in \mathcal{T}$ can be constructed in $O(m_u \log m_u)$ time. Therefore, we can construct $\mathcal{T}$ in $O(n \log^2 n)$ time by Property 2.

Recall that for $i, j \in [1..n]$ with $i < j$, it holds $w'(i, j) = \mathsf{OPT}(i, j)$. We give an outline of the algorithm that computes $\mathsf{OPT}(i, j)$ only for the non-confluent flow model since a similar argument holds even for the confluent flow model with minor modification. The main task is to find a value z^* that minimizes $\Phi^{i,j}(z)$, i.e., $\mathsf{OPT}(i, j) = \Phi^{i,j}(z^*)$. By Lemma 2, such the value z^* is the pseudo-intersection point of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$ on $[W_i, W_{j-1}]$.

Before explaining our algorithms, we need introduce the following definition:

Definition 2. *For integers $i, \ell, r \in [1..n]$ with $i < \ell \leq r$, we denote by $\theta^{i,+,[\ell..r]}(z)$ the upper envelope of functions $\{\theta^{i,+,h}(z) \mid h \in [\ell..r]\}$, that is,*

$$\theta^{i,+,[\ell..r]}(z) = \max\{\theta^{i,+,h}(z) \mid h \in [\ell..r]\}.$$

For integers $i, \ell, r \in [1..n]$ with $\ell \leq r < i$, we denote by $\theta^{i,-,[\ell..r]}(z)$ the upper envelope of functions $\{\theta^{i,-,h}(z) \mid h \in [\ell..r]\}$, that is,

$$\theta^{i,-,[\ell..r]}(z) = \max\{\theta^{i,-,h}(z) \mid h \in [\ell..r]\}.$$

Algorithm for computing $\mathsf{OPT}(i, j)$ for given $i, j \in [1..n]$ with $i < j$

Phase 1: Find a set U of the maximal subpath nodes for $P_{i+1,j-1}$ by walking on segment tree $\mathcal{T}$ from leaf $i + 1$ to leaf $j - 1$.

Phase 2: For each $u \in U$, compute a real interval $\mathcal{I}_u^+$ such that $\theta^{i,+}(z) = \theta^{i,+,[\ell_u..r_u]}(z)$ holds on any $z \in \mathcal{I}_u^+$, and a real interval $\mathcal{I}_u^-$ such that $\theta^{j,-}(z) = \theta^{j,-,[\ell_u..r_u]}(z)$ holds on any $z \in \mathcal{I}_u^-$, both of which are obtained by using information stored at node u. See Sect. 5.1 in [14] for the details.

Phase 3: Compute the pseudo-intersection point z^* of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$ on $[W_i, W_{j-1}]$ by using real intervals obtained in Phase 2. See Sect. 5.2 in [14] for the details.

Phase 4: Compute $\mathsf{OPT}(i, j) = \Phi^{i,j}(z^*)$ as follows: By formula (7), we have

$$\begin{aligned}\Phi^{i,j}(z^*) &= \int_0^{z^*} \theta^{i,+}(t)dt + \int_{z^*}^{W_n} \theta^{j,-}(t)dt \\ &= \sum_{u \in U} \left\{ \int_{\mathcal{I}_u^+ \cap [0,z^*]} \theta^{i,+,[\ell_u..r_u]}(t)dt + \int_{\mathcal{I}_u^- \cap [z^*,W_n]} \theta^{j,-,[\ell_u..r_u]}(t)dt \right\}.\end{aligned}$$

For each $u \in U$, we compute integrals $\int \theta^{i,+,[\ell_u..r_u]}(t)dt$ and $\int \theta^{j,-,[\ell_u..r_u]}(t)dt$ by using the information stored at u. See Sect. 5.3 in [14] for the details.

For the cases of $i = 0$ or $j = n + 1$, we can also compute $w'(0, j) = \Phi^{j,-}(0)$ and $w'(i, n + 1) = \Phi^{i,+}(W_n)$ by the same operations except for Phase 3.

We give the following lemma about the running time of the above algorithm for the case of general edge capacities. See Lemma 8 in [14] for the proof.

Lemma 5 (Key lemma for general capacity). *Let us suppose that a segment tree $\mathcal{T}$ is available. Given two integers $i, j \in [0..n+1]$ with $i < j$, one can compute a value $w'(i, j)$ in $O(\log^3 n)$ time for the confluent/non-confluent flow model.*

Recalling that the running time for construction of data structure $\mathcal{T}$ is $O(n \log^2 n)$, Lemmas 3, 4 and 5 imply the following main theorem.

Theorem 1 (Main theorem for general capacity). *Given a dynamic flow path network $\mathcal{P}$, there exists an algorithm that finds an optimal k-sink under the confluent/non-confluent flow model in time* $\min\{O(kn \log^3 n), n2^{O(\sqrt{\log k \log\log n})} \log^3 n\}$.

When the capacities of $\mathcal{P}$ are uniform, we can improve the running time for computing $w'(i, j)$ to $O(\log^2 n)$ time with minor modification. See Lemma 9 in [14] for the proof.

Lemma 6 (Key lemma for uniform capacity). *Let us suppose that a segment tree $\mathcal{T}$ is available. Given two integers $i, j \in [1..n]$ with $i < j$, one can compute a value $w'(i, j)$ in $O(\log^2 n)$ time for the confluent/non-confluent flow model when the capacities are uniform.*

Theorem 2 (Main theorem for uniform capacity). *Given a dynamic flow path network $\mathcal{P}$ with a uniform capacity, there exists an algorithm that finds an optimal k-sink under the confluent/non-confluent flow model in time* $\min\{O(kn \log^2 n), n2^{O(\sqrt{\log k \log\log n})} \log^2 n\}$.

4 Data Structures Associated with Nodes of $\mathcal{T}$

In the rest of the paper, we introduce novel data structures associated with each node u of segment tree $\mathcal{T}$, which are used to compute $\mathsf{OPT}(i, j)$ in $O(\text{poly} \log n)$ time. Note that our data structures generalize the *capacities and upper envelopes tree (CUE tree)* provided by Bhattacharya et al. [7].

Recall the algorithm for computing $\mathsf{OPT}(i, j)$ shown in Sect. 3. To explain the data structures, let us see more precisely how the algorithm performs in Phase 2. Confirm that for $z \in [W_i, W_{j-1}]$, it holds $\theta^{i,+}(z) = \max\{\theta^{i,+,[\ell_u..r_u]}(z) \mid u \in U\}$, where U is a set of the maximal subpath nodes for $P_{i+1,j-1}$. Let us focus on function $\theta^{i,+,[\ell_u..r_u]}(z)$ for a node $u \in U$ only on interval $(W_{\ell_u-1}, W_n]$ since it holds $\theta^{i,+,[\ell_u..r_u]}(z) = 0$ if $z \leq W_{\ell_u-1}$. Interval $(W_{\ell_u-1}, W_n]$ consists of three

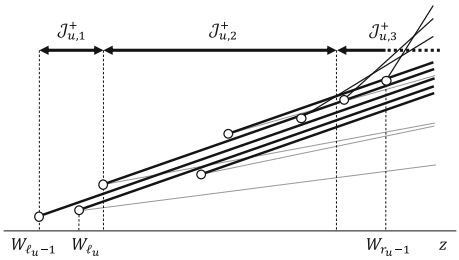

Fig. 2. Illustration of $\mathcal{J}^+_{u,1}$, $\mathcal{J}^+_{u,2}$ and $\mathcal{J}^+_{u,3}$. The thick half lines have the same slope of $1/C_{i,\ell_u}$, the gray half lines have slopes $\leq 1/C_{i,\ell_u}$, and the regular half lines have slopes $> 1/C_{i,\ell_u}$. The upper envelope of all the thick half lines and the regular half lines is function $\theta^{i,+,[\ell_u..r_u]}(z)$.

left-open-right-closed intervals $\mathcal{J}^+_{u,1}$, $\mathcal{J}^+_{u,2}$ and $\mathcal{J}^+_{u,3}$ that satisfy the following conditions: (i) For $z \in \mathcal{J}^+_{u,1}$, $\theta^{i,+,[\ell_u..r_u]}(z) = \theta^{i,+,\ell_u}(z)$. (ii) For $z \in \mathcal{J}^+_{u,2}$, $\theta^{i,+,[\ell_u..r_u]}(z) = \theta^{i,+,[\ell_u+1..r_u]}(z)$ and its slope is $1/C_{i,\ell_u}$. (iii) For $z \in \mathcal{J}^+_{u,3}$, $\theta^{i,+,[\ell_u..r_u]}(z) = \theta^{i,+,[\ell_u+1..r_u]}(z)$ and its slope is greater than $1/C_{i,\ell_u}$. See also Fig. 2. Thus in Phase 2, the algorithm computes $\mathcal{J}^+_{u,1}$, $\mathcal{J}^+_{u,2}$ and $\mathcal{J}^+_{u,3}$ for all $u \in U$, and combines them one by one to obtain intervals $\mathcal{I}^+_u$ for all $u \in U$. To implement these operations efficiently, we construct some data structures at each node u of $\mathcal{T}$. To explain the data structures stored at u, we introduce the following definition:

Definition 3. *For integers $i, \ell, r \in [1..n]$ with $i < \ell \leq r$ and a positive real c, let $\bar{\theta}^{i,+,[\ell..r]}(c,z) = \max\{\bar{\theta}^{i,+,h}(c,z) \mid h \in [\ell..r]\}$, where*

$$\bar{\theta}^{i,+,j}(c,z) = \begin{cases} 0 & \text{if } z \leq W_{j-1}, \\ \frac{z-W_{j-1}}{c} + \tau \cdot L_{i,j} & \text{if } z > W_{j-1}. \end{cases} \tag{10}$$

For integers $i, \ell, r \in [1..n]$ with $\ell \leq r < i$ and a positive real c, let $\bar{\theta}^{i,-,[\ell..r]}(c,z) = \max\{\bar{\theta}^{i,-,h}(c,z) \mid h \in [\ell..r]\}$, where

$$\bar{\theta}^{i,-,j}(c,z) = \begin{cases} \frac{W_j-z}{c} + \tau \cdot L_{j,i} & \text{if } z < W_j, \\ 0 & \text{if } z \geq W_j. \end{cases} \tag{11}$$

We can see that for $z \in \mathcal{J}^+_{u,2}$, $\theta^{i,+,[\ell_u+1..r_u]}(z) = \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(C_{i,\ell_u}, z) + \tau \cdot L_{i,\ell_u}$, and for $z \in \mathcal{J}^+_{u,3}$, $\theta^{i,+,[\ell_u+1..r_u]}(z) = \theta^{\ell_u,+,[\ell_u+1..r_u]}(z) + \tau \cdot L_{i,\ell_u}$. We then store at u of $\mathcal{T}$ the information for computing in $O(\text{poly}\log n)$ time $\theta^{\ell_u,+,[\ell_u+1..r_u]}(z)$ for any $z \in [0, W_n]$ as TYPE I, and also one for computing in $O(\text{poly}\log n)$ time $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ for any $c > 0$ and any $z \in [0, W_n]$ as TYPE III.

In Phase 4, the algorithm requires computing integrals $\int_0^z \theta^{\ell_u,+,[\ell_u+1..r_u]}(t)dt$ for any $z \in [0, W_n]$, and $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt$ for any $c > 0$ and any $z \in [0, W_n]$, for which the information is stored at each $u \in \mathcal{T}$ as TYPEs II and IV, respectively.

In a symmetric manner, we also store at each $u \in \mathcal{T}$ the information for computing $\theta^{r_u,-,[\ell_u..r_u-1]}(z)$, $\int_z^{W_n} \theta^{r_u,-,[\ell_u..r_u-1]}(t)dt$, $\bar{\theta}^{r_u,-,[\ell_u..r_u-1]}(c,z)$, and $\int_z^{W_n} \bar{\theta}^{r_u,-,[\ell_u..r_u-1]}(c,t)dt$ as TYPEs I, II, III, and IV, respectively.

Let us introduce what information is stored as TYPEs I–IV at $u \in \mathcal{T}$. See Sect. 4 in [14] for the detail.

TYPE I. We give the information only for computing $\theta^{\ell_u,+,[\ell_u+1..r_u]}(z)$ stored at $u \in \mathcal{T}$ as TYPE I since the case for $\theta^{r_u,-,[\ell_u..r_u-1]}(z)$ is symmetric. By Definition 2, the function $\theta^{\ell_u,+,[\ell_u+1..r_u]}(z)$ is the upper envelope of m_u functions $\theta^{\ell_u,+,h}(z)$ for $h \in [\ell_u + 1..r_u]$. Let $\mathcal{B}^{u,+} = (b_1^{u,+} = 0, b_2^{u,+}, \ldots, b_{N^{u,+}}^{u,+} = W_n)$ denote a sequence of breakpoints of $\theta^{\ell_u,+,[\ell_u+1..r_u]}(z)$, where $N^{u,+}$ is the number of breakpoints. For each $p \in [1..N^{u,+} - 1]$, let $H_p^{u,+} \in [\ell_u + 1..r_u]$ such that $\theta^{\ell_u,+,[\ell_u+1..r_u]}(z) = \theta^{\ell_u,+,H_p^{u,+}}(z)$ holds for any $z \in (b_p^{u,+}, b_{p+1}^{u,+}]$. As TYPE I, each node $u \in \mathcal{T}$ is associated with following two lists:

1. Pairs of breakpoint $b_p^{u,+}$ and value $\theta^{\ell_u,+,[\ell_u+1..r_u]}(b_p^{u,+})$, and
2. Pairs of range $(b_p^{u,+}, b_{p+1}^{u,+}]$ and index $H_p^{u,+}$.

Note that the above lists can be constructed in $O(m_u \log m_u)$ time for each $u \in \mathcal{T}$. By Property 2, we can obtain the whole information of TYPE I of $\mathcal{T}$ in time $O(n \log^2 n)$. We now give the application of TYPE I. See Lemma 11 in [14] for the proof.

Lemma 7 (Query with TYPE I). *Suppose that TYPE I of $\mathcal{T}$ is available. Given a node $u \in \mathcal{T}$ and a real value $z \in [0, W_n]$, we can obtain*

(i) index $H \in [\ell_u + 1..r_u]$ such that $\theta^{\ell_u,+,H}(z) = \theta^{\ell_u,+,[\ell_u+1..r_u]}(z)$, and
(ii) index $H \in [\ell_u..r_u - 1]$ such that $\theta^{r_u,-,H}(z) = \theta^{r_u,-,[\ell_u..r_u-1]}(z)$

in time $O(\log n)$ respectively. Furthermore, if the capacities of $\mathcal{P}$ are uniform and $z \notin [W_{\ell_u}, W_{r_u-1}]$, we can obtain the above indices in time $O(1)$.

TYPE II. We give the information only for computing $\int_0^z \theta^{\ell_u,+,[\ell_u+1..r_u]}(t)dt$ stored at $u \in \mathcal{T}$ as TYPE II since the case for $\int_z^{W_n} \theta^{r_u,-,[\ell_u..r_u-1]}(t)dt$ is symmetric. Each node $u \in \mathcal{T}$ contains a list of all pairs of breakpoint $b_p^{u,+}$ and value $\int_0^{b_p^{u,+}} \theta^{\ell_u,+,[\ell_u+1..r_u]}(t)dt$. This list can be constructed in $O(m_u)$ time for each $u \in \mathcal{T}$ by using TYPE I of u. By Property 2, we obtain the whole information of TYPE II of $\mathcal{T}$ in time $O(n \log n)$. We give the application of TYPEs I and II. See Lemma 12 in [14] for the proof.

Lemma 8 (Query with TYPEs I and II). *Suppose that TYPEs I and II of $\mathcal{T}$ is available. Given a node $u \in \mathcal{T}$ and a real value $z \in [0, W_n]$, we can obtain (i) value $\int_0^z \theta^{\ell_u,+,[\ell_u+1..r_u]}(t)dt$, and (ii) value $\int_z^{W_n} \theta^{r_u,-,[\ell_u..r_u-1]}(t)dt$ in time $O(\log n)$ respectively. Furthermore, if the capacities of $\mathcal{P}$ are uniform and $z \notin [W_{\ell_u}, W_{r_u-1}]$, we can obtain the above values in time $O(1)$.*

TYPE III. We give the information only for computing $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ stored at $u \in \mathcal{T}$ as TYPE III since the case for $\bar{\theta}^{r_u,-,[\ell_u..r_u-1]}(c,z)$ is symmetric. Note that it is enough to prepare for the case of $z \in (W_{\ell_u}, W_{r_u}]$ since it holds that $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z) = 0$ for $z \in [0, W_{\ell_u}]$ and

$$\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z) = \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,W_{r_u}) + \frac{z - W_{r_u}}{c}$$

for $z \in (W_{r_u}, W_n]$, of which the first term is obtained by prepared information with $z = W_{r_u}$ and the second term is obtained by elementally calculation.

For each $u \in \mathcal{T}$, we construct a *persistent segment tree* as TYPE III. Referring to formula (10), each function $\bar{\theta}^{\ell_u,+,j}(c,z)$ for $j \in [l_u+1..r_u]$ is linear in $z \in (W_{j-1}, W_n]$ with the same slope $1/c$. Let us make parameter c decrease from ∞ to 0, then all the slopes $1/c$ increase from 0 to ∞. As c decreases, the number of subfunctions that consist of $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ also decreases one by one from m_u to 1. Let $c_h^{u,+}$ be a value c at which the number of subfunctions of $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ becomes $m_u - h$ while c decreases. Note that we have $\infty = c_0^{u,+} > c_1^{u,+} > \cdots > c_{m_u-1}^{u,+} > 0$. Let us define indices $j_1^h, \ldots, j_{m_u-h}^h$ with $l_u + 1 = j_1^h < \cdots < j_{m_u-h}^h \leq r_u$ corresponding to the subfunctions of $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c_h^{u,+},z)$, that is, for any integer $p \in [1..m_u - h]$, we have

$$\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c_h^{u,+},z) = \bar{\theta}^{\ell_u,+,j_p^h}(c_h^{u,+},z) \quad \text{if } z \in (W_{j_p^h-1}, W_{j_{p+1}^h-1}], \tag{12}$$

where $j_{m_u-h+1}^h - 1 = r_u$. We give the following lemma about the property of $c_h^{u,+}$. See Lemma 13 in [14] for the proof.

Lemma 9. *For each node $u \in \mathcal{T}$, all values $c_1^{u,+}, \ldots, c_{m_u-1}^{u,+}$ can be computed in $O(m_u \log m_u)$ time.*

By the above argument, while $c \in (c_h^{u,+}, c_{h-1}^{u,+}]$ with some $h \in [1..m_u]$ (where $c_{m_u}^{u,+} = 0$), the representation of $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ (with $m_u - h + 1$ subfunctions) remains the same. Our fundamental idea is to consider segment trees corresponding to each interval $(c_h^{u,+}, c_{h-1}^{u,+}]$ with $h \in [1..m_u]$, and construct a persistent data structure for such the segment trees.

First of all, we introduce a segment tree T_h with root ρ_h to compute $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ for $c \in (c_h^{u,+}, c_{h-1}^{u,+}]$ with $h \in [1..m_u]$. Tree T_h contains m_u leaves labeled as $l_u + 1, \ldots, r_u$. Each leaf j corresponds to interval $(W_{j-1}, W_j]$. For a node $\nu \in T_h$, let ℓ_ν (resp. r_ν) denote the label of the leftmost (resp. rightmost) leaf of the subtree rooted at ν. Let p_ν denote the parent of ν if $\nu \neq \rho_h$. We say a node $\nu \in T_h$ *spans* an interval $(W_{\ell_\nu-1}, W_{r_\nu}]$. For some two integers $i, j \in [\ell_u + 1..r_u]$ with $i < j$, if $(W_{\ell_\nu-1}, W_{r_\nu}] \subseteq (W_{i-1}, W_j]$ and $(W_{\ell_{p_\nu}-1}, W_{r_{p_\nu}}] \not\subseteq (W_{i-1}, W_j]$, then ν is called a *maximal subinterval node* for $(W_{i-1}, W_j]$. A segment tree T_h satisfies the following property similar to Property 1: For any two integers $i, j \in [\ell_u + 1..r_u]$ with $i < j$, the number of maximal subinterval nodes in T_h for $(W_{i-1}, W_j]$ is $O(\log m_u)$. For each $p \in [1..m_u - h + 1]$,

we store function $\bar{\theta}^{\ell_u,+,j_p^{h-1}}(c,z)$ at all the maximal subinterval nodes for interval $(W_{j_p^{h-1}-1}, W_{j_{p+1}^{h-1}-1}]$, which takes $O(m_u \log m_u)$ time by the property. The other nodes in T_h contains NULL.

If we have T_h, for given $z \in (W_{\ell_u}, W_{r_u}]$ and $c \in (c_h^{u,+}, c_{h-1}^{u,+}]$, we can compute value $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ in time $O(\log m_u)$ as follows: Starting from root ρ_h, go down to a child such that its spanned interval contains z until we achieve a node that contains some function $\bar{\theta}^{\ell_u,+,j}(c,z)$ (not NULL). Now, we know $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z) = \bar{\theta}^{\ell_u,+,j}(c,z)$, which can be computed by elementally calculation.

If we explicitly construct T_h for all $h \in [1..m_u]$, it takes $O(m_u^2 \log m_u)$ time for each node $u \in \mathcal{T}$, which implies by Property 2 that $O(n^2 \log^2 n)$ time is required in total. However, using the fact that T_h and T_{h+1} are almost same except for at most $O(\log m_u)$ nodes, we can construct a persistent segment tree in $O(m_u \log m_u)$ time, in which we can search as if all of T_h are maintained. Thus, we obtain the whole information of TYPE III of $\mathcal{T}$ in time $O(n \log^2 n)$ by Property 2.

Using this persistent segment tree, we can compute $\bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$ for any $z \in [0, W_n]$ and any $c > 0$ in $O(\log m_u)$ time as follows: Find integer h over $[1..m_u]$ such that $c \in (c_h^{u,+}, c_{h-1}^{u,+}]$ in $O(\log m_u)$ time by binary search, and then search in the persistent segment tree as T_h in time $O(\log m_u)$.

Lemma 10 (Query with TYPE III). *Suppose that TYPE III of $\mathcal{T}$ is available. Given a node $u \in \mathcal{T}$, real values $z \in [0, W_n]$ and $c > 0$, we can obtain*

(i) index $H \in [\ell_u + 1..r_u]$ such that $\bar{\theta}^{\ell_u,+,H}(c,z) = \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,z)$, and
(ii) index $H \in [\ell_u..r_u - 1]$ such that $\bar{\theta}^{r_u,-,H}(c,z) = \bar{\theta}^{r_u,-,[\ell_u..r_u-1]}(c,z)$ in time $O(\log n)$ respectively.

TYPE IV. We give the information only for computing $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt$ stored at $u \in \mathcal{T}$ as TYPE IV since the case for $\int_z^{W_n} \bar{\theta}^{\ell_u,-,[\ell_u..r_u-1]}(c,t)dt$ is symmetric. Similar to TYPE III, we prepare only for the case of $z \in (W_{\ell_u}, W_{r_u}]$ since it holds that $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt = 0$ for $z \in [0, W_{\ell_u}]$ and

$$\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt = \int_0^{W_{r_u}} \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt + \frac{(z - W_{r_u})^2}{2c}$$

for $z \in (W_{r_u}, W_n]$, of which the first term can be obtained by prepared information with $z = W_{r_u}$ and the second term by elementally calculation.

For each $u \in \mathcal{T}$, we construct a persistent segment tree again, which is similar to one shown in the previous section. To begin with, consider the case of $c \in (c_h^{u,+}, c_{h-1}^{u,+}]$ with some $h \in [1..m_u]$ (where recall that $c_0^{u,+} = \infty$ and $c_{m_u}^{u,+} = 0$), and indices $j_1^{h-1}, \cdots, j_{m_u-h+1}^{h-1}$ that satisfy (12). In this case, for $z \in (W_{j_p^{h-1}-1}, W_{j_{p+1}^{h-1}-1}]$ with $p \in [1..m_u - h + 1]$, we have

$$\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt = \sum_{q=1}^{p-1} \left\{ \int_{W_{j_q^{h-1}-1}}^{W_{j_{q+1}^{h-1}-1}} \bar{\theta}^{\ell_u,+,j_q^{h-1}}(c,t)dt \right\} + \int_{W_{j_p^{h-1}-1}}^{z} \bar{\theta}^{\ell_u,+,j_p^{h-1}}(c,t)dt. \tag{13}$$

For ease of reference, we use $F^{h,p}(c,z)$ instead of the right hand side of (13).

Similarly to the explanation for TYPE III, let T_h be a segment tree with root ρ_h and m_u leaves labeled as $l_u+1,\ldots,r_u$, and each leaf j of T_h corresponds to interval $(W_{j-1},W_j]$. In the same manner as for TYPE III, for each $p \in [1..m_u-h+1]$, we store function $F^{h,p}(c,z)$ at all the maximal subinterval nodes in T_h for interval $(W_{j_p^{h-1}-1},W_{j_{p+1}^{h-1}-1}]$. Using T_h, for any $z \in (W_{\ell_u},W_{r_u}]$ and any $c \in (c_h^{u,+},c_{h-1}^{u,+}]$, we can compute value $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt$ in time $O(\log m_u)$ by summing up all functions of nodes on a path from root ρ_h to leaf with an interval that contains z. Actually, we store functions in a more complicated way in order to maintain them as a persistent data structure. We construct a persistent segment tree at $u \in \mathcal{T}$ in $O(m_u \log m_u)$ time, in which we can search as if all of T_h are maintained. Using this persistent segment tree, we can compute $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt$ for any $z \in [0,W_n]$ and any $c>0$ in $O(\log m_u)$ time in the same manner as for TYPE III.

Lemma 11 (Query with TYPE IV). *Suppose that TYPE IV of $\mathcal{T}$ is available. Given a node $u \in \mathcal{T}$, real values $z \in [0,W_n]$ and $c>0$, we can obtain (i) value $\int_0^z \bar{\theta}^{\ell_u,+,[\ell_u+1..r_u]}(c,t)dt$, and (ii) value $\int_z^{W_n} \bar{\theta}^{r_u,-,[\ell_u..r_u-1]}(c,t)dt$ in time $O(\log n)$ respectively.*

5 Conclusion

We remark here that our algorithms can be extended to the minsum k-sink problem in a dynamic flow path network, in which each vertex v_i has the cost λ_i for locating a sink at v_i, and we minimize $\mathsf{AT}(\mathcal{P},\mathbf{x},\mathbf{d})+\sum_i\{\lambda_i \mid \mathbf{x} \text{ consists of } v_i\}$. Then, the same reduction works with link costs $w''(i,j)=w'(i,j)+\lambda_i$, which still satisfy the concave Monge property. This implies that our approach immediately gives algorithms of the same running time.

Acknowledgement. Yuya Higashikawa: Supported by JSPS Kakenhi Grant-in-Aid for Young Scientists (20K19746), JSPS Kakenhi Grant-in-Aid for Scientific Research (B) (19H04068), and JST CREST (JPMJCR1402).
Naoki Katoh: Supported by JSPS Kakenhi Grant-in-Aid for Scientific Research (B) (19H04068), and JST CREST (JPMJCR1402).
Junichi Teruyama: Supported by JSPS Kakenhi Grant-in-Aid for Scientific Research (B) (19H04068), and JST CREST (JPMJCR1402).

References

1. Alstrup, S., Gavoille, C., Kaplan, H., Rauhe, T.: Nearest common ancestors: a survey and a new distributed algorithm. In: Proceedings of the the 14th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 258–264 (2002)
2. Belmonte, R., Higashikawa, Y., Katoh, N., Okamoto, Y.: Polynomial-time approximability of the k-sink location problem. CoRR abs/1503.02835 (2015)

3. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000). https://doi.org/10.1007/10719839_9
4. Benkoczi, R., Bhattacharya, B., Higashikawa, Y., Kameda, T., Katoh, N.: Minsum k-Sink problem on dynamic flow path networks. In: Iliopoulos, C., Leong, H.W., Sung, W.-K. (eds.) IWOCA 2018. LNCS, vol. 10979, pp. 78–89. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94667-2_7
5. Benkoczi, R., Bhattacharya, B., Higashikawa, Y., Kameda, T., Katoh, N.: Minsum k-sink problem on path networks. Theor. Comput. Sci. **806**, 388–401 (2020)
6. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Heidelberg (2010)
7. Bhattacharya, B., Golin, M.J., Higashikawa, Y., Kameda, T., Katoh, N.: Improved algorithms for computing k-Sink on dynamic flow path networks. In: Ellen, F., Kolokolova, A., Sack, J.R. (eds.) WADS 2017. LNCS, vol. 10389, pp. 133–144. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_12
8. Chen, D., Golin, M.J.: Sink evacuation on trees with dynamic confluent flows. In: 27th International Symposium on Algorithms and Computation (ISAAC 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
9. Chen, D., Golin, M.J.: Minmax centered k-partitioning of trees and applications to sink evacuation with dynamic confluent flows. CoRR abs/1803.09289 (2018)
10. Ford, L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. Oper. Res. **6**(3), 419–433 (1958)
11. Higashikawa, Y.: Studies on the space exploration and the sink location under incomplete information towards applications to evacuation planning. Ph.D. thesis, Kyoto University, Japan (2014)
12. Higashikawa, Y., Golin, M.J., Katoh, N.: Minimax regret sink location problem in dynamic tree networks with uniform capacity. J. Graph Algorithms Appl. **18**(4), 539–555 (2014)
13. Higashikawa, Y., Golin, M.J., Katoh, N.: Multiple sink location problems in dynamic path networks. Theor. Comput. Sci. **607**, 2–15 (2015)
14. Higashikawa, Y., Katoh, N., Teruyama, J., Watase, K.: Almost linear time algorithms for minsum k-sink problems on dynamic flow path networks (a full version of the paper). CoRR abs/2010.05729 (2020)
15. Hoppe, B., Tardos, E.: The quickest transshipment problem. Math. Oper. Res. **25**(1), 36–62 (2000)
16. Mamada, S., Uno, T., Makino, K., Fujishige, S.: An $O(n \log^2 n)$ algorithm for a sink location problem in dynamic tree networks. Discret. Appl. Math. **154**, 2387–2401 (2006)
17. Schieber, B.: Computing a minimum weight k-link path in graphs with the concave monge property. J. Algorithms **29**(2), 204–222 (1998)
18. Skutella, M.: An introduction to network flows over time. In: Cook, W., Lovász, L., Vygen, J. (eds.) Research Trends in Combinatorial Optimization, pp. 451–482. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-76796-1_21