

Chapter 5

Almost Linear Time Algorithms for Some Problems on Dynamic Flow Networks



Yuya Higashikawa, Naoki Katoh, and Junichi Teruyama

Abstract Motivated by evacuation planning, several problems regarding *dynamic flow networks* have been studied in recent years. A dynamic flow network consists of an undirected graph with positive edge lengths, positive edge capacities, and positive vertex weights. The road network in an area can be treated as a graph where the edge lengths are the distances along the roads and the vertex weights are the number of people at each site. An edge capacity limits the number of people that can enter the edge per unit time. In a dynamic flow network, when particular points on edges or vertices called *sinks* are given, all of the people are required to evacuate from the vertices to the sinks as quickly as possible. This chapter gives an overview of two of our recent results on the problem of locating multiple sinks in a dynamic flow path network such that the max/sum of evacuation times for all the people to sinks is minimized, and we focus on techniques that enable the problems to be solved in almost linear time.

5.1 Introduction

Recently, many parts of the world have been affected by disasters including earthquakes, nuclear plant accidents, volcanic eruptions, and flooding, highlighting the urgent need for orderly evacuation planning. One powerful tool 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 a positive length and capacity.

Y. Higashikawa (✉) · N. Katoh · J. Teruyama
University of Hyogo, Kobe, Japan
e-mail: higashikawa@gsis.u-hyogo.ac.jp

N. Katoh
e-mail: naoki.katoh@gsis.u-hyogo.ac.jp

J. Teruyama
e-mail: junichi.teruyama@gsis.u-hyogo.ac.jp

© The Author(s) 2022
N. Katoh et al. (eds.), *Sublinear Computation Paradigm*,
https://doi.org/10.1007/978-981-16-4095-7_5

Table 5.1 Summary of minmax k -sink problems

Path	General capacities: $O(n \log n + k^2 \log^4 n)$, $O(n \log^3 n)$ [7]
	Uniform capacity: $O(n + k^2 \log^2 n)$, $O(n \log n)$ [7]
Tree	General capacities: $O(\max\{k, \log n\} \cdot kn \log^4 n)$ [9]
	Uniform capacity: $O(\max\{k, \log n\} \cdot kn \log^3 n)$ [9]
General graph	General capacities: FPTAS for a fixed k [3]
	Uniform capacity: FPTAS for a fixed k [3]

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*, in which the objective is to send exactly the right amount of supply out of sources into sinks while satisfying demand constraints in the minimum overall time. Hoppe and Tardos [17] 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 is still an open problem. Readers are referred to a recent survey by Skutella [20] on dynamic flows.

This chapter discusses related problems called k -sink problems [3, 5–9, 14–16, 18], in which the objective is to find the locations of k sinks in a given dynamic flow network so that all the supply is sent to the sinks as quickly as possible. The following two criteria can be naturally considered for determining the optimality of the locations: minimization of *evacuation completion time* and *aggregate evacuation time* (i.e., *average evacuation time*). We call the k -sink problem that requires finding the locations of k sinks that minimize the evacuation completion time (resp., the aggregate evacuation time) the *minmax* (resp., *minsum*) k -sink problem. Although several papers have studied minmax k -sink problems in dynamic flow networks [3, 7–9, 14, 15, 18], minsum k -sink problems in dynamic flow networks have not been studied except for the case of path networks [5, 6, 15, 16].¹ Tables 5.1 and 5.2 summarize the previous results for the minmax k -sink problems and the minsum k -sink problems, respectively.

There are two models for the evacuation method. 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, almost all of the papers that deal with the k -sink problems [3, 5–9, 15] adopt the confluent flow model, while only one paper [16] handles both of the models.

Although it may seem natural to model the evacuation behavior of people by treating each supply as a discrete quantity as in [17, 18], almost all of the previous papers on sink problems [3, 7–9, 14–16] have treated each supply as a continuous

¹ Note that the minsum 1-sink problem in general networks can be solved in polynomial time by applying the following two facts: (1) Baumann and Skutella [2] provided a polynomial time algorithm for the problem of computing a dynamic flow to a fixed sink in a general network while minimizing the aggregate evacuation time. (2) For the minsum 1-sink problem in general networks, one can prove that there exists an optimal sink located at a vertex in a similar manner to the well-known *node optimality theorem* for the 1-median problem [12].

Table 5.2 Summary of minsum k -sink problems

Path	General capacities: $O(kn \log^4 n)$ [6] $\min\{O(kn \log^3 n), n2^{O(\sqrt{\log k \log \log n})} \log^3 n\}$ [16]
	Uniform capacity: $O(kn \log^3 n)$ [5] $\min\{O(kn \log^2 n), n2^{O(\sqrt{\log k \log \log n})} \log^2 n\}$ [16]
Tree	Open
General graph	

quantity since it is easier to treat the problems mathematically and the effect is negligible when the number of people is large. Throughout this chapter, we adopt the model with continuous supplies.

We also give an overview of two of our recent results [7, 16] on the problems of locating multiple sinks on dynamic flow path networks such that the max/sum of evacuation times for all the people to sinks is minimized, and we focus on algorithmic frameworks that enable solving the problems in almost linear time.

5.2 Preliminaries

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 $\langle w_1, \dots, w_n \rangle$ of which each component w_i is the *weight* of vertex v_i representing the amount of supply (e.g., the number of evacuees or cars) located at v_i , \mathbf{c} is a vector $\langle c_1, \dots, c_{n-1} \rangle$ of which each component c_i is the *capacity* of edge e_i representing the upper bound on the flow amount that can enter e_i per unit time, \mathbf{l} is a vector $\langle \ell_1, \dots, \ell_{n-1} \rangle$ of which each component ℓ_i is the *length* of edge e_i (i.e., the distance between two end vertices of e_i), and τ is the time taken for unit supply to move unit distance along any edge.

We say that 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 $e \in E$. We assume that path P can be represented by a horizontal line segment along which the vertices v_1, v_2, \dots, v_n are arranged in order from left to right. For two points $p, q \in P$, $p < q$ means that p lies to the left side of q . For two points $p, q \in P$, $p \leq q$ means that $p < q$ or p and q lie at the same location. For two points $p, q \in P$ such that $p \leq q$, p divides an edge (v_i, v_{i+1}) in the ratio $r_p : 1 - r_p$, and q divides an edge (v_j, v_{j+1}) in the ratio $r_q : 1 - r_q$, let $L(p, q)$ be the distance between p and q , that is, $L(p, q) = (1 - r_p)\ell_i + r_q\ell_j + \sum_{h=i+1}^{j-1} \ell_h$ (where $\sum_{h=i+1}^i \ell_h = 0$ and $\sum_{h=i+1}^{i-1} \ell_h = -\ell_i$). 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 , and by $\mathcal{P}_{i,j}$ a subnetwork

of \mathcal{P} consisting of subpaths $P_{i,j}$. Let $L_{i,j}$ be the distance between v_i and v_j , that is, $L_{i,j} = \sum_{h=i}^{j-1} \ell_h$, and let $C_{i,j}$ be the minimum capacity among all the edges between v_i and v_j , that is, $C_{i,j} = \min\{c_h \mid h \in [i..j-1]\}$. 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, 4].

A k -sink \mathbf{x} is a k -tuple (x_1, \dots, x_k) of points on P such that $x_i < x_j$ for any $i < j$. We assume that no two sinks lie on the same edge.² We define the function Id for point $p \in P$ as follows: the value $\text{Id}(p)$ is an integer such that $v_{\text{Id}(p)} \leq p < v_{\text{Id}(p)+1}$ holds, that is, if p lies on edge (v_i, v_{i+1}) or at vertex v_i , $\text{Id}(p) = i$. A *divider* \mathbf{d} is a $(k-1)$ -tuple (d_1, \dots, d_{k-1}) of real values such that $0 \leq d_i < d_j \leq W_n$ for any $i < j$. A pair (\mathbf{x}, \mathbf{d}) is called *valid* if and only if $W_{\text{Id}(x_i)} \leq d_i \leq W_{\text{Id}(x_{i+1})}$ holds for any i . A valid pair (\mathbf{x}, \mathbf{d}) determines what amount of supply from which vertex flows to which sink so that the portion $d_i - d_{i-1}$ of supply is assigned to flow to sink x_i , where $d_0 = 0$ and $d_k = W_n$. More precisely, given a valid pair (\mathbf{x}, \mathbf{d}) , the portion $W_{\text{Id}(x_i)} - d_{i-1}$ of supply that originates from the left side of x_i flows to sink x_i , and the portion $d_i - W_{\text{Id}(x_i)}$ of 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 $h \in [1..n]$, the portion $d_i - W_{h-1}$ of the w_h supply at v_h flows to sink x_i and the rest of the $W_h - d_i$ supply flows to 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 $\{W_1, \dots, W_n\}$, whereas the non-confluent flow model does not. For a dynamic flow path network \mathcal{P} and a valid pair (\mathbf{x}, \mathbf{d}) , the *evacuation completion time* $\text{CT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is the time at which all the supply completes the evacuation. The *aggregate evacuation time* $\text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is the sum of the evacuation completion time for all the supply. Explicit definitions of these are given in Sect. 5.3.

5.3 Objective Functions

Suppose that we are given a divider $\mathbf{d} = (d_1, \dots, d_{k-1})$. This \mathbf{d} implies that we have k 1-sink subproblems. The i th subproblem consists of a subnetwork $\mathcal{P}_{h,h'}$ such that the weight of v_j is w_j for $j \in [h+1..h'-1]$, while those of v_h and $v_{h'}$ are $W_h - d_{i-1}$ and $d_i - W_{h'-1}$, respectively, where $W_{h-1} < d_{i-1} \leq W_h$ and $W_{h'-1} < d_i \leq W_{h'}$. To explicitly define the evacuation completion time and the aggregate evacuation time,

² It turns out that this assumption does not result in a loss of generality once the cost function is introduced later. If some adjacent two sinks x_i and x_{i+1} lie on edge (v_j, v_{j+1}) , that is, $v_j < x_i < x_{i+1} < v_{j+1}$, moving x_i to v_j or moving x_{i+1} to v_{j+1} does not increase the cost.

we first consider the case of the 1-sink problem, and then extend the argument to the general case of the k -sink problem.

5.3.1 Objective Functions for the 1-Sink Problem

Given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices, we assign a unique sink to a point x , that is, $\mathbf{x} = (x)$ and $\mathbf{d} = ()$, which is the 0-tuple. We consider only the case where x is on an edge e_i excluding its end vertices, that is, $v_i < x < v_{i+1}$, since the case where x is on a vertex can be treated similarly. In this case, all the supply on the left side of x (i.e., at v_1, \dots, v_i) flows to the right toward sink x , and all the supply on the right side of x (i.e., at v_{i+1}, \dots, v_n) flows to the left toward sink x .

To treat this case, we introduce some new notation. Let the function $\theta^{x,+}(z)$ denote the time at which the first $z - W_i$ of supply on the right side of x completes its evacuation to sink x (where $\theta^{x,+}(z) = 0$ for $z \in [0, W_i]$). Similarly, let $\theta^{x,-}(z)$ denote the time at which the first $W_i - z$ of supply on the left side of x completes its evacuation to sink x (where $\theta^{x,-}(z) = 0$ for $z \in [W_i, W_n]$). Higashikawa [13] showed that the values $\theta^{x,+}(W_n)$ and $\theta^{x,-}(0)$, which are the evacuation completion times for all the supply on the right and left sides of x , respectively, are given by the following formulae:

$$\theta^{x,+}(W_n) = \max \left\{ \frac{W_n - W_{j-1}}{C_{i,j}} + \tau \cdot L(x, v_j) \mid j \in [i+1..n] \right\}, \text{ and} \quad (5.1)$$

$$\theta^{x,-}(0) = \max \left\{ \frac{W_j}{C_{j,i+1}} + \tau \cdot L(v_j, x) \mid j \in [1..i] \right\}. \quad (5.2)$$

Using these, the evacuation completion time $\text{CT}(\mathcal{P}, (x), ())$ is given by

$$\text{CT}(\mathcal{P}, (x), ()) = \max \{ \theta^{x,+}(W_n), \theta^{x,-}(0) \}. \quad (5.3)$$

We can generalize formulae (5.1) and (5.2) to the case of any $z \in [0, W_n]$ as follows:

$$\theta^{x,+}(z) = \max \{ \theta^{x,+,j}(z) \mid j \in [i+1..n] \}, \quad (5.4)$$

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

$$\theta^{x,+,j}(z) = \begin{cases} 0 & \text{if } z \leq W_{j-1}, \\ \frac{z - W_{j-1}}{C_{i,j}} + \tau \cdot L(x, v_j) & \text{if } z > W_{j-1}, \end{cases} \quad (5.5)$$

and

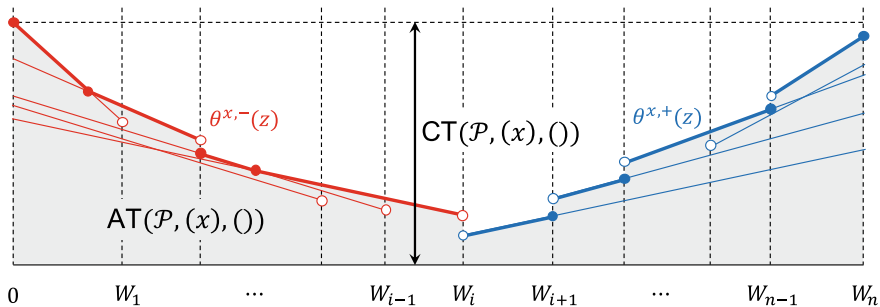


Fig. 5.1 The blue (resp., red) thick half-open segments indicate the function $\theta^{x,+}(z)$ (resp., $\theta^{x,-}(z)$). The gray area indicates $\text{AT}(\mathcal{P}, (x), ())$

$$\theta^{x,-}(z) = \max\{\theta^{x,-,j}(z) \mid j \in [1..i]\}, \quad (5.6)$$

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

$$\theta^{x,-,j}(z) = \begin{cases} \frac{W_j - z}{C_{j,i+1}} + \tau \cdot L(v_j, x) & \text{if } z < W_j, \\ 0 & \text{if } z \geq W_j. \end{cases} \quad (5.7)$$

Then, the aggregate evacuation times for the supply on the right and left sides of x are

$$\int_{W_i}^{W_n} \theta^{x,+}(z) dz \quad \text{and} \quad \int_0^{W_i} \theta^{x,-}(z) dz,$$

respectively. Thus, the aggregate evacuation time $\text{AT}(\mathcal{P}, (x), ())$ is given by

$$\text{AT}(\mathcal{P}, (x), ()) = \int_0^{W_i} \theta^{x,-}(z) dz + \int_{W_i}^{W_n} \theta^{x,+}(z) dz. \quad (5.8)$$

See also Fig. 5.1.

5.3.2 Objective Functions for k -Sink

Let us consider a valid pair consisting of a k -sink $\mathbf{x} = (x_1, \dots, x_k)$ and a divider $\mathbf{d} = (d_1, \dots, d_{k-1})$ such that each sink is on an edge excluding its end vertices, that is, $v_{\text{Id}(x_i)} < x_i < v_{\text{Id}(x_i)+1}$. In this situation, for each $i \in [1..k]$, the first $d_i - W_{\text{Id}(x_i)}$ of supply on the right side of x_i and the first $W_{\text{Id}(x_i)} - d_{i-1}$ of supply on the left side of x_i move to sink x_i . By the argument of the previous section, the evacuation completion times for the supply on the right and left sides of x_i are represented by

$$\theta^{x_i,+}(d_i) \text{ and } \theta^{x_i,-}(d_{i-1}),$$

respectively. Thus, the evacuation completion time $\text{CT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is given by

$$\text{CT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) = \max \{ \theta^{x_i,+}(d_i), \theta^{x_i,-}(d_{i-1}) \mid i \in [1..k] \}, \quad (5.9)$$

where $d_0 = 0$ and $d_k = W_n$. The aggregate evacuation times for the supply on the right and left sides of x_i are

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

respectively. Thus, the aggregate evacuation time $\text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is given by

$$\text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) = \sum_{i \in [1..k]} \left(\int_{d_{i-1}}^{W_{\text{Id}(x_i)}} \theta^{x_i,-}(z) dz + \int_{W_{\text{Id}(x_i)}}^{d_i} \theta^{x_i,+}(z) dz \right), \quad (5.10)$$

where $d_0 = 0$ and $d_k = W_n$.

5.4 Minmax k -Sink Problems on Paths

In this section, we consider the minmax k -sink problems on path networks under the confluent flow model, which is precisely defined as

(MINMAX- k -SINK-PATH-CONFLUENT-FLOW)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$.

Goal: Find a solution (\mathbf{x}, \mathbf{d}) to the problem

$$\begin{aligned} \text{min.} \quad & \text{CT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) \\ \text{s.t.} \quad & \mathbf{x} = (x_1, \dots, x_k) \in P^k, & x_h < x_l \quad \forall h < l, \\ & \mathbf{d} = (d_1, \dots, d_{k-1}) \in \{W_h \mid h \in [1..n]\}^{k-1}, & W_{\text{Id}(x_h)} \leq d_h \leq W_{\text{Id}(x_{h+1})} \quad \forall h. \end{aligned}$$

For the MINMAX- k -SINK-PATH-CONFLUENT-FLOW problem, [7] reported the following result, which is the best so far:

Theorem 5.1 ([7]) *The MINMAX- k -SINK-PATH-CONFLUENT-FLOW problem can be solved in $O(\min\{n \log n + k^2 \log^4 n, n \log^3 n\})$ time. Moreover, if the capacities of \mathcal{P} are uniform, the MINMAX- k -SINK-PATH-CONFLUENT-FLOW problem can be solved in $O(\min\{n + k^2 \log^2 n, n \log n\})$ time.*

Theorem 5.1 implies that the problem is solved in almost linear time for any k . In [7], two kinds of algorithms are provided: One is an $O(n \log n + k^2 \log^4 n)$ time

algorithm based on the *parametric search method*, and the other is an $O(n \log^3 n)$ time algorithm based on the *sorted matrix method*.

Both algorithms require repeatedly solving the problems of locating 1-sink for multiple choices of different subnetworks. Note that the optimal solution for the problem of locating 1-sink on $\mathcal{P}_{i,j}$ is a point x^* that minimizes the following expression over $x \in P_{i,j}$

$$\text{CT}(\mathcal{P}_{i,j}, (x), ()) = \max \{ \theta^{x,+}(W_j), \theta^{x,-}(W_{i-1}) \}. \quad (5.11)$$

Both algorithms also require repeatedly performing *feasibility tests* for multiple choices of different subnetworks. We say that $\mathcal{P}_{i,j}$ is (t, q) -feasible if and only if the answer of the following decision problem is “yes”:

(FEASIBILITY-TEST-FOR-SUBPATH)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$, a positive real $t \in \mathbb{R}^+$, integers q, i, j satisfying $q \in [1..k]$ and $i, j \in [1..n]$ with $i < j$.

Goal: Determine whether there exists a pair of vectors $(\mathbf{x}', \mathbf{d}')$ such that

$$\begin{aligned} \text{CT}(\mathcal{P}_{i,j}, \mathbf{x}', \mathbf{d}') &\leq t, \\ \mathbf{x}' &= (x_1, \dots, x_q) \in P_{i,j}^q, & x_h < x_l \quad \forall h < l, \\ \mathbf{d}' &= (d_1, \dots, d_{q-1}) \in \{W_h \mid h \in [i..j]\}^{q-1}, & W_{\text{Id}(x_h)} \leq d_h \leq W_{\text{Id}(x_{h+1})} \quad \forall h. \end{aligned}$$

Note that [7] developed a data structure called the *CUE tree* to efficiently compute $\theta^{x,-}(W_{i-1})$ and $\theta^{x,+}(W_j)$ for any integers $i, j \in [1..n]$ with $i < j$ and any $x \in P_{i,j}$. For the case of general edge capacities, the CUE tree can be constructed in $O(n \log n)$ time, and $\theta^{x,-}(W_{i-1})$ and $\theta^{x,+}(W_j)$ can be computed in $O(\log^2 n)$ time by using the CUE tree. See [7] for more detail.

Lemma 5.1 ([7]) *Given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices, the CUE tree can be constructed in $O(n \log n)$ time. Moreover, if the capacities of \mathcal{P} are uniform, the CUE tree can be constructed in $O(n)$ time.*

Lemma 5.2 ([7]) *Given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices, suppose that the CUE tree is available. Then, for any integers $i, j \in [1..n]$ with $i < j$ and any $x \in P_{i,j}$, $\theta^{x,-}(W_{i-1})$ and $\theta^{x,+}(W_j)$ can be computed in $O(\log^2 n)$ time. Moreover, if the capacities of \mathcal{P} are uniform, $\theta^{x,-}(W_{i-1})$ and $\theta^{x,+}(W_j)$ can be computed in $O(\log n)$ time.*

In the rest of this section, we first describe how feasibility tests are performed in Sect. 5.4.1 and how the 1-sink problem for a subnetwork is solved in Sect. 5.4.2, and then show the frameworks of the parametric search method in Sect. 5.4.3 and the sorted matrix method in Sect. 5.4.4.

5.4.1 Feasibility Test

In [7], to solve the MINMAX- k -SINK-PATH-CONFLUENT-FLOW problem, an algorithm repeatedly tests the (t, q) -feasibility of $\mathcal{P}_{i,j}$ for multiple choices of different 4-tuples (t, q, i, j) . Let $\text{CT}_{\text{OPT}}(q, i, j)$ denote the optimal cost for the problem of locating q -sink on $\mathcal{P}_{i,j}$. Then, for a positive real $t \in \mathbb{R}^+$, integers q, i, j satisfying $q \in [1..k]$ and $i, j \in [1..n]$ with $i < j$, $\mathcal{P}_{i,j}$ is (t, q) -feasible if and only if $\text{CT}_{\text{OPT}}(q, i, j) \leq t$ holds.

Lemma 5.3 ([7]) *Given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices, suppose that the CUE tree is available. For integers q, i, j satisfying $q \in [1..k]$ and $i, j \in [1..n]$ with $i < j$, the (t, q) -feasibility of $\mathcal{P}_{i,j}$ can be tested in $O(\min\{n \log^2 n, k \log^3 n\})$ time. Moreover, if the capacities of \mathcal{P} are uniform, the (t, q) -feasibility of $\mathcal{P}_{i,j}$ can be tested in $O(\min\{n, k \log n\})$ time.*

Proof We prove only the case of general capacities. For the case of uniform capacity, see [7].

To determine the (t, q) -feasibility of $\mathcal{P}_{i,j}$, we first place the sinks consecutively from left to right as far to the right as possible. We then compute the maximum integer h such that $\theta^{v_h, -}(W_{i-1}) \leq t$ and $\theta^{v_{h+1}, -}(W_{i-1}) > t$ holds. Next, we solve

$$\theta^{v_{h+1}, -}(W_{i-1}) - \alpha \cdot \tau \ell_h = t \quad (5.12)$$

for α . If $\alpha < 1$, we move the leftmost sink x_1 to the point that divides edge $e_h = (v_h, v_{h+1})$ at a ratio of $1 - \alpha : \alpha$, otherwise we place x_1 at v_h . We then compute the maximum integer l_1 such that $\theta^{x_1, +}(W_{l_1}) \leq t$ and $\theta^{x_1, +}(W_{l_1+1}) > t$ holds. We thus determine the maximal subnetwork \mathcal{P}_{i,l_1} such that $\text{CT}_{\text{OPT}}(1, i, l_1) \leq t$. In the same manner, we repeatedly isolate the maximal subnetworks $\mathcal{P}_{i,l_1}, \mathcal{P}_{l_1+1,l_2}, \mathcal{P}_{l_2+1,l_3}, \dots$, and if the q th subnetwork is found to have $l_q < j$, then $\mathcal{P}_{i,j}$ is not (t, q) -feasible, otherwise it is (t, q) -feasible.

Let us now look at the time complexity. Isolating \mathcal{P}_{i,l_1} consists of (a) computing h , (b) solving the equation for α , and (c) computing l_1 . Obviously (b) takes $O(1)$ time. For (a), applying a binary search takes $O(\log^3 n)$ time because we compute $\theta^{v_a, -}(W_{i-1})$ over $a \in [i..j]$ $O(\log n)$ times and each $\theta^{v_a, -}(W_{i-1})$ can be computed in $O(\log^2 n)$ time using the CUE tree by Lemma 5.2. Similarly (c) takes $O(\log^3 n)$ time by binary search. In this way, we can isolate at most q subnetworks in $O(q \log^3 n) = O(k \log^3 n)$ time. However, if we simply scan from left to right instead of using a binary search for (a) and (c), that is, if we compute $\theta^{v_a, -}(W_{i-1})$ for $a = i, i + 1, \dots, h, h + 1$ and $\theta^{x_b, -}(W_b)$ for $b = h + 1, h + 2, \dots, l_1, l_1 + 1$, it takes $O((l_1 - i) \log^2 n)$ time to determine \mathcal{P}_{i,l_1} . In this way, we can isolate at most p subnetworks in $O((j - i) \log^2 n) = O(n \log^2 n)$ time. \square

5.4.2 Solving the 1-Sink Problem

Lemma 5.4 ([7]) *Given a dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices, suppose that the CUE tree is available. For any integers i, j satisfying $i, j \in [1..n]$ with $i < j$, $\text{CT}_{\text{OPT}}(1, i, j)$ can be computed in $O(\log^3 n)$ time. Moreover, if the capacities of \mathcal{P} are uniform, $\text{CT}_{\text{OPT}}(1, i, j)$ can be computed in $O(\log n)$ time.*

Proof We prove only the case of general capacities. See [7] for the case of uniform capacity.

Recalling Eq. (5.11), we have

$$\begin{aligned} \text{CT}_{\text{OPT}}(1, i, j) &= \min_{x \in P_{i,j}} \text{CT}(\mathcal{P}_{i,j}, (x), ()) \\ &= \min_{x \in P_{i,j}} \max \{ \theta^{x,+}(W_j), \theta^{x,-}(W_{i-1}) \}. \end{aligned} \quad (5.13)$$

Because $\theta^{x,+}(W_j)$ and $\theta^{x,-}(W_{i-1})$ are monotonically decreasing and monotonically increasing, respectively, in $x \in P_{i,j}$, if an integer $h \in [i..j]$ satisfies $\theta^{v_h,-}(W_{i-1}) \leq \theta^{v_h,+}(W_j)$ and $\theta^{v_{h+1},-}(W_{i-1}) > \theta^{v_{h+1},+}(W_j)$, then there exists x^* that minimizes $\text{CT}(\mathcal{P}_{i,j}, (x), ())$ on edge e_h including v_h and v_{h+1} . We can apply binary search to compute this h , which can be done in $O(\log^3 n)$ time using the CUE tree (see Lemma 5.2). Once h is determined, x^* can be computed as follows: We solve

$$\theta^{v_{h+1},-}(W_{i-1}) - \alpha \cdot \tau \ell_h = \theta^{v_h,+}(W_j) - (1 - \alpha) \cdot \tau \ell_h \quad (5.14)$$

for α in $O(1)$ time. If $\alpha \leq 0$, let $x^* = v_{h+1}$ and compute $\text{CT}_{\text{OPT}}(1, i, j) = \text{CT}(\mathcal{P}_{i,j}, (v_{h+1}), ())$. If $\alpha \geq 1$, let $x^* = v_h$ and compute $\text{CT}_{\text{OPT}}(1, i, j) = \text{CT}(\mathcal{P}_{i,j}, (v_h), ())$. Otherwise, let x^* be the point that divides edge $e_h = (v_h, v_{h+1})$ at a ratio of $1 - \alpha : \alpha$ and compute $\text{CT}_{\text{OPT}}(1, i, j) = \theta^{v_{h+1},-}(W_{i-1}) - \alpha \cdot \tau \ell_h = \theta^{v_h,+}(W_j) - (1 - \alpha) \cdot \tau \ell_h$. Using the CUE tree, we can compute these values in $O(\log^2 n)$ time. Thus, $\text{CT}_{\text{OPT}}(1, i, j)$ can be computed in $O(\log^3 n) + O(1) + O(\log^2 n) = O(\log^3 n)$ time. \square

5.4.3 Parametric Search Method

In the parametric search method, we first compute the maximum integer i_1 such that $\mathcal{P}_{i_1+1,n}$ is not $(\text{CT}_{\text{OPT}}(1, 1, i_1), k - 1)$ -feasible and store $t_1 = \text{CT}_{\text{OPT}}(1, 1, i_1 + 1)$ as a feasible value. Note that $t^* = \text{CT}_{\text{OPT}}(k, 1, n)$ satisfies $\text{CT}_{\text{OPT}}(1, 1, i_1) < t^* \leq t_1$. To compute i_1 , we apply binary search by executing $O(\log n)$ tests for $(\text{CT}_{\text{OPT}}(1, 1, a), k - 1)$ -feasibility of $\mathcal{P}_{a+1,n}$ over $1 \leq a \leq n$. For an integer a , we can compute $\text{CT}_{\text{OPT}}(1, 1, a)$ in $O(\log^3 n)$ time by Lemma 5.4. Also, by Lemma 5.3, we can test whether $\mathcal{P}_{a+1,n}$ is $(\text{CT}_{\text{OPT}}(1, 1, a), k - 1)$ -feasible in $O(k \log^3 n)$ time. Summarizing these arguments, we can compute i_1 and t_1 in $\{O(\log^3 n) + O(k \log^3 n)\}$

$\times O(\log n) = O(k \log^4 n)$ time. Next, we compute the maximum integer i_2 such that $\mathcal{P}_{i_2+1,n}$ is not $(\text{CT}_{\text{OPT}}(1, i_1 + 1, i_2), k - 2)$ -feasible and store $t_2 = \text{CT}_{\text{OPT}}(1, i_1 + 1, i_2 + 1)$ as a feasible value, which can be done in $O(k \log^4 n)$ time in the same manner as in the computation of (i_1, t_1) . Sequentially, we determine $(i_3, t_3), \dots, (i_{k-1}, t_{k-1})$ in $(k - 3) \times O(k \log^4 n)$ time and eventually compute $t_k = \text{CT}_{\text{OPT}}(1, i_{k-1} + 1, n)$ in $O(\log^3 n)$ time. Note that $t^* = \min\{t_i \mid i = 1, 2, \dots, k\}$ holds, which can be computed in $O(k)$ time. We then execute a (t^*, k) -feasibility test for \mathcal{P} in $O(k \log^3 n)$ time, so that the optimal k -sink is obtained. We thus see that the problem can be solved in $(k - 1) \times O(k \log^4 n) + O(\log^3 n) + O(k) + O(k \log^3 n) = O(k^2 \log^4 n)$ time once the CUE tree is constructed. Since it takes $O(n \log n)$ time to construct the CUE tree by Lemma 5.1, the total time complexity is $O(n \log n + k^2 \log^4 n)$.

For the case of uniform capacity, the same argument holds. Applying Lemmas 5.1, 5.3, and 5.4, we have a total time complexity of $O(n + k^2 \log^2 n)$.

5.4.4 Sorted Matrix Method

A matrix A is *sorted* if and only if each row and column of A is sorted in non-decreasing order. The sorted matrix method is based on the following lemma shown in [11]:

Lemma 5.5 ([11]) *Consider a minimization problem Q with an instance \mathcal{I} of size n . Suppose that the feasibility of any value for \mathcal{I} can be tested in $g(n)$ time. Let A be an $n \times n$ sorted matrix such that each element can be computed in $f(n)$ time. Then, the minimum element of A that is feasible for Q can be found in $O(nf(n) + g(n) \log n)$ time.*

In [7], an $n \times n$ matrix A is defined such that the (i, j) th entry of A is given by

$$A[i, j] = \begin{cases} \text{CT}_{\text{OPT}}(1, n - i + 1, j) & \text{if } n - i + 1 \leq j \\ 0 & \text{otherwise.} \end{cases} \quad (5.15)$$

Note that we do not actually compute all the elements of A , but compute the element $A[i, j]$ on demand as needed.

Let us confirm that matrix A is sorted. It is also clear that matrix A includes $\text{CT}_{\text{OPT}}(1, l, r)$ for every pair of integers (l, r) such that $l, r \in [1..n]$ with $l < r$. In addition, there exists a pair (l, r) such that $\text{CT}_{\text{OPT}}(1, l, r) = \text{CT}_{\text{OPT}}(k, 1, n)$. These facts imply that the minimum element $A[i, j]$ such that \mathcal{P} is $(A[i, j], k)$ -feasible is $\text{CT}_{\text{OPT}}(k, 1, n)$, and hence we can apply Lemma 5.5 to solve the MINMAX- k -SINK-PATH-CONFLUENT-FLOW problem as follows: Once the CUE tree is constructed, we have $f(n) = O(\log^3 n)$ by Lemma 5.4 and $g(n) = O(n \log^2 n)$ by Lemma 5.3, so the problem can be solved in $O(n \log^3 n)$ time. Because it takes $O(n \log n)$ time to construct the CUE tree by Lemma 5.1, the total time complexity is $O(n \log n) + O(n \log^3 n) = O(n \log^3 n)$.

For the case of uniform capacity, the same argument holds. Applying Lemmas 5.1, 5.3, and 5.4, we have a total time complexity of $O(n \log n)$.

5.5 Minsum k -Sink Problems on Paths

In this section, our task is to find a valid pair (\mathbf{x}, \mathbf{d}) that minimizes the aggregate evacuation time $\text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$. This task can be precisely represented as follows:

(MINSUM- k -SINK-PATH)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$.

Goal: Find a solution (\mathbf{x}, \mathbf{d}) to the problem

$$\begin{aligned} \min. \quad & \text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) \\ \text{s.t.} \quad & \mathbf{x} = (x_1, \dots, x_k) \in P^k, & x_h < x_l \quad \forall h < l, \\ & \mathbf{d} = (d_1, \dots, d_{k-1}) \in \mathbb{R}^{k-1}, & W_{\text{Id}(x_h)} \leq d_h \leq W_{\text{Id}(x_{h+1})} \quad \forall h. \end{aligned}$$

(MINSUM- k -SINK-PATH-CONFLUENT-FLOW)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$.

Goal: Find a solution (\mathbf{x}, \mathbf{d}) to the problem

$$\begin{aligned} \min. \quad & \text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) \\ \text{s.t.} \quad & \mathbf{x} = (x_1, \dots, x_k) \in P^k, & x_h < x_l \quad \forall h < l, \\ & \mathbf{d} = (d_1, \dots, d_{k-1}) \in \{W_h \mid h \in [1..n]\}^{k-1}, & W_{\text{Id}(x_h)} \leq d_h \leq W_{\text{Id}(x_{h+1})} \quad \forall h. \end{aligned}$$

For the MINSUM- k -SINK-PATH problem, [16] reported the following result, which is the best so far:

Theorem 5.2 ([16]) *The MINSUM- k -SINK-PATH/MINSUM- k -SINK-PATH-CONFLUENT-FLOW problems can be solved in $\min\{O(kn \log^3 n), n2^{O(\sqrt{\log k \log \log n})} \log^3 n\}$ time. Moreover, if the capacities of \mathcal{P} are uniform, then both the problems can be solved in $\min\{O(kn \log^2 n), n2^{O(\sqrt{\log k \log \log n})} \log^2 n\}$ time.*

For the confluent flow model, it was shown in [6, 15] that for the minsum k -sink problems, there exists an optimal k -sink such that all of the k sinks are at vertices. [16] extended this fact to the non-confluent flow model.

Lemma 5.6 ([6, 15, 16]) *For the minsum k -sink problem in a dynamic flow path network, there exists an optimal k -sink such that all of the k sinks are at vertices under the confluent/non-confluent flow model.*

Lemma 5.6 implies that it is sufficient to consider only the case where every sink is at a vertex. Thus, we suppose $\mathbf{x} = (x_1, \dots, x_k) \in V^k$, where $x_i < x_j$ for $i < j$.

The fundamental idea of [16] for solving the MINSUM- k -SINK-PATH problem is to reduce it to the *minimum k -link path problem*. In the minimum k -link path problem, 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 a weight $w'(i, j)$. A k -link path is a path that contains exactly k edges. The task is to find a k -link path from v'_1 to v'_n that minimizes the sum of weights of k edges. The minimum k -link path problem is represented as follows:

(MINIMUM- k -LINK-PATH)

Input: A weighted complete DAG $G = (V', E', w')$.

Goal: Find a k -link path $(v'_{a_0} = v'_1, v'_{a_1}, v'_{a_2}, \dots, v'_{a_{k-1}}, v'_{a_k} = v'_n)$ from v'_1 to v'_n .

$$\begin{aligned} \min. \quad & \sum_{i=1}^k w'(a_{i-1}, a_i) \\ \text{s.t.} \quad & a_i \in [0..n], \quad a_0 = 1, a_k = n, a_h < a_l \quad \forall h < l. \end{aligned}$$

Schieber [19] showed that the MINIMUM- k -LINK-PATH can be solved in almost linear time³ regardless of k if the weight function w' satisfies the *concave Monge property*.

Definition 5.1 (*Concave Monge property*) We say that a function $f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}$ 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 5.7 ([19]) *Given a weighted complete DAG with n vertices, if the weight function satisfies the concave Monge property, the MINIMUM- k -LINK-PATH can be solved in $\min\{O(kn), n2^{O(\sqrt{\log k \log \log n})}\}$ time.*

Higashikawa et al. [16] presented a reduction from MINSUM- k -SINK-PATH to MINIMUM- $(k + 1)$ -LINK-PATH such that the weight function w' satisfies the concave Monge property. Let a dynamic flow path network $\mathcal{P} = (P = (V, E), \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$ with n vertices be an instance of MINSUM- k -SINK-PATH. 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

$$w'(i, j) = \begin{cases} \text{AT}_{\text{OPT}}(i, j) & i, j \in [1..n], i < j, \\ \text{AT}(\mathcal{P}_{i,n}, (v_i), () & i \in [1..n] \text{ and } j = n + 1, \\ \text{AT}(\mathcal{P}_{1,j}, (v_j), () & i = 0 \text{ and } j \in [1..n], \\ \infty & i = 0 \text{ and } j = n + 1, \end{cases} \quad (5.16)$$

where $\text{AT}_{\text{OPT}}(i, j)$ is the optimal aggregate evacuation time required to move all the supply between v_i and v_j to one of two sinks v_i or v_j . On the weighted complete DAG G constructed as above, let us consider a $(k + 1)$ -link path $(v'_{a_0} =$

³ Note that we assume that the weight function w' is not given explicitly, but that a value $w'(i, j)$ can be obtained in constant time whenever required.

$v'_0, v'_{a_1}, \dots, v'_{a_k}, v'_{a_{k+1}} = v'_{n+1}$) from v'_0 to v'_{n+1} , where a_1, \dots, a_k are integers satisfying $0 < a_1 < a_2 < \dots < a_k < n + 1$. The sum of weights of this $(k + 1)$ -link path is

$$\sum_{i=0}^k w'(a_i, a_{i+1}) = \text{AT}(\mathcal{P}_{1,a_1}, (v_{a_1}), ()) + \sum_{i=1}^{k-1} \text{AT}_{\text{OPT}}(a_i, a_{i+1}) + \text{AT}(\mathcal{P}_{a_k,n}, (v_{a_k}), ()).$$

This value is equivalent to $\min_{\mathbf{d}} \text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ for a k -sink $\mathbf{x} = (v_{a_1}, v_{a_2}, \dots, v_{a_k})$, 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} .

Let us consider the following subtasks:

(MINSUM-FLOW-FOR-SUBPATH)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$, integers $i, j \in [1..n]$ with $i < j$.

Goal: Find a value d such that

$$\begin{array}{ll} \min. & \text{AT}(\mathcal{P}_{i,j}, \mathbf{x} = (v_i, v_j), \mathbf{d} = (d)) \\ \text{s.t.} & W_i \leq d \leq W_{j-1}. \end{array}$$

(MINSUM-FLOW-FOR-SUBPATH-CONFLUENT-FLOW)

Input: A dynamic flow path network $\mathcal{P} = (P, \mathbf{w}, \mathbf{c}, \mathbf{l}, \tau)$, integers $i, j \in [1..n]$ with $i < j$.

Goal: Find a value d such that

$$\begin{array}{ll} \min. & \text{AT}(\mathcal{P}_{i,j}, \mathbf{x} = (v_i, v_j), \mathbf{d} = (d)) \\ \text{s.t.} & d \in \{W_h \mid h \in [i..j - 1]\}. \end{array}$$

Note that [16] developed a data structure to efficiently solve both of these problems. This data structure can be constructed in $O(n \log^2 n)$ time and can be used to solve MINSUM-FLOW-FOR-SUBPATH/MINSUM-FLOW-FOR-SUBPATH-CONFLUENT-FLOW in $O(\log^3 n)$ time. See [16] for details.

Lemma 5.8 ([16]) *For a given dynamic flow path network \mathcal{P} with n vertices, there exists a segment tree \mathcal{T} that satisfies the following conditions:*

1. \mathcal{T} can be constructed in $O(n \log^2 n)$ time.
2. MINSUM-FLOW-FOR-SUBPATH/MINSUM-FLOW-FOR-SUBPATH-CONFLUENT-FLOW can be solved in $O(\log^3 n)$ time by using \mathcal{T} .
3. If the capacities of \mathcal{P} are uniform, then MINSUM-FLOW-FOR-SUBPATH/MINSUM-FLOW-FOR-SUBPATH-CONFLUENT-FLOW can be solved in $O(\log^2 n)$ time by using \mathcal{T} .

Because w' satisfies the concave Monge property (see Sect. 5.5.2), Lemmas 5.7 and 5.8 lead to Theorem 5.2.

In the rest of this section, we first observe the properties of the aggregate evacuation time in Sect. 5.5.1 and then show that the weighted function w' obtained by the reduction satisfies the concave Monge property in Sect. 5.5.2.

5.5.1 Property of Aggregate Evacuation Time

Recalling that we consider only the case where every sink is at a vertex, we simply use $\theta^{i,+}(z)$ and $\theta^{i,-}(z)$ instead of $\theta^{v_i,+}(z)$ and $\theta^{v_i,-}(z)$, respectively.

We next give the general form of the aggregate evacuation time. Let $\phi^{i,+}(z)$ denote 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

$$\begin{aligned} \phi^{i,+}(z) &= \int_{W_i}^z \theta^{i,+}(t) dt = \int_0^z \theta^{i,+}(t) dt \quad \text{and} \\ \phi^{i,-}(z) &= \int_z^{W_{i-1}} \theta^{i,-}(t) dt = \int_z^{W_n} \theta^{i,-}(t) dt = - \int_{W_n}^z \theta^{i,-}(t) dt \end{aligned} \tag{5.17}$$

(see Fig. 5.2). For $i, j \in [1..n]$ with $i < j$, we define

$$\phi^{i,j}(z) = \phi^{i,+}(z) + \phi^{j,-}(z) = \int_0^z \theta^{i,+}(t) dt + \int_z^{W_n} \theta^{j,-}(t) dt \tag{5.18}$$

for $z \in [W_i, W_{j-1}]$.

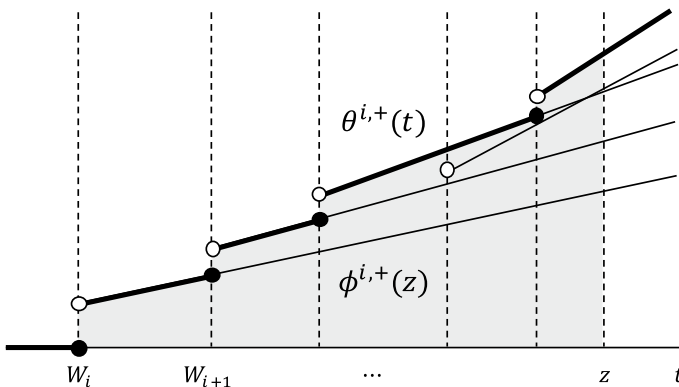


Fig. 5.2 Thick half-open segments represent the function $\theta^{i,+}(t)$ and the gray area represents $\phi^{i,+}(z)$ for some $z > W_i$

Suppose that we are given a k -sink $\mathbf{x} = (x_1, \dots, x_k) \in V^k$ and a divider $\mathbf{d} = (d_1, \dots, d_{k-1})$. Recalling the definition of $\text{Id}(p)$ for $p \in P$, we have $x_i = v_{\text{Id}(x_i)}$ for all $i \in [1..k]$. Because each sink is at a vertex, by simply modifying the integration intervals in Eq. (5.10), the aggregate evacuation time $\text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d})$ is given by

$$\begin{aligned} \text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) &= \sum_{i \in [1..k]} \left(\int_{d_{i-1}}^{W_{\text{Id}(x_i)-1}} \theta^{x_i, -}(z) dz + \int_{W_{\text{Id}(x_i)}}^{d_i} \theta^{x_i, +}(z) dz \right) \\ &= \sum_{i \in [1..k]} \left(\int_{d_{i-1}}^{W_{\text{Id}(x_i)-1}} \theta^{\text{Id}(x_i), -}(z) dz + \int_{W_{\text{Id}(x_i)}}^{d_i} \theta^{\text{Id}(x_i), +}(z) dz \right). \end{aligned} \quad (5.19)$$

By Eqs. (5.17), (5.18) and (5.19), we have

$$\begin{aligned} \text{AT}(\mathcal{P}, \mathbf{x}, \mathbf{d}) &= \sum_{i \in [1..k]} \left(\int_{d_{i-1}}^{W_{\text{Id}(x_i)-1}} \theta^{\text{Id}(x_i), -}(z) dz + \int_{W_{\text{Id}(x_i)}}^{d_i} \theta^{\text{Id}(x_i), +}(z) dz \right) \\ &= \sum_{i \in [1..k]} (\phi^{\text{Id}(x_i), -}(d_{i-1}) + \phi^{\text{Id}(x_i), +}(d_i)) \\ &= \phi^{\text{Id}(x_1), -}(0) + \sum_{i \in [1..k-1]} \phi^{\text{Id}(x_i), \text{Id}(x_{i+1})}(d_i) + \phi^{\text{Id}(x_k), +}(W_n). \end{aligned} \quad (5.20)$$

In the rest of this section, we show the important properties of $\phi^{i,j}(z)$. Let us first confirm that by Eq. (5.17), 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 , and therefore $\phi^{i,j}(z)$ is convex in z . We have a more useful lemma that gives the conditions for the minimizer of $\phi^{i,j}(z)$.

Lemma 5.9 ([16]) *For any $i, j \in [1..n]$ with $i < j$, there uniquely exists*

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

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

Proof By Eqs. (5.4) and (5.5), $\theta^{i,+}(z)$ is strictly increasing in $z \in [W_i, W_n]$. Similarly, by Eqs. (5.6) and (5.7), $\theta^{j,-}(z)$ is strictly decreasing in $z \in [0, W_{j-1}]$. Thus, there uniquely exists $z^* \in [W_i, W_{j-1}]$.

We then see that for any $z' \in [W_i, z^*]$,

$$\begin{aligned} \phi^{i,j}(z^*) - \phi^{i,j}(z') &= \phi^{i,+}(z^*) + \phi^{j,-}(z^*) - (\phi^{i,+}(z') + \phi^{j,-}(z')) \\ &= \int_{z'}^{z^*} \theta^{i,+}(t) dt - \int_{z'}^{z^*} \theta^{j,-}(t) dt \\ &= \int_{z'}^{z^*} \{\theta^{i,+}(t) - \theta^{j,-}(t)\} dt \leq 0, \end{aligned}$$

and for any $z'' \in [z^*, W_{j-1}]$,

$$\begin{aligned} \phi^{i,j}(z^*) - \phi^{i,j}(z'') &= \phi^{i,+}(z^*) + \phi^{j,-}(z^*) - (\phi^{i,+}(z'') + \phi^{j,-}(z'')) \\ &= \int_{z''}^{z^*} \theta^{i,+}(t) dt - \int_{z''}^{z^*} \theta^{j,-}(t) dt \\ &= - \int_{z^*}^{z''} \{\theta^{i,+}(t) - \theta^{j,-}(t)\} dt \leq 0, \end{aligned}$$

which imply that z^* minimizes $\phi^{i,j}(z)$ on $[W_i, W_{j-1}]$. \square

In the following sections, this z^* is called the *pseudo-intersection point*⁴ of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$.

5.5.2 Concave Monge Property

We now show that the function w' defined in Eq. (5.16) satisfies the concave Monge property under the non-confluent flow model. We omit the proof for the confluent flow model, since the proof can be constructed similarly to the one for the confluent flow model. See [16] for details.

Let us give some observations of $\text{AT}_{\text{OPT}}(i, j)$. Under the non-confluent flow model, for any $i, j \in [1..n]$ with $i < j$, $\text{AT}_{\text{OPT}}(i, j) = \min_{z \in [W_i, W_{j-1}]} \phi^{i,j}(z)$. Lemma 5.9 implies that $\phi^{i,j}(z)$ on $[W_i, W_{j-1}]$ is minimized when z is the pseudo-intersection point of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$. For any $i, j \in [1..n]$ with $i < j$, let $\alpha^{i,j}$ denote the pseudo-intersection point of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$.

Thus, we have

$$\text{AT}_{\text{OPT}}(i, j) = \phi^{i,j}(\alpha^{i,j}) = \int_0^{\alpha^{i,j}} \theta^{i,+}(z) dz + \int_{\alpha^{i,j}}^{W_n} \theta^{j,-}(z) dz. \quad (5.21)$$

We give the following two lemmas.

Lemma 5.10 ([16]) *For any integer $i \in [1..n - 1]$ and any $z \in [0, W_n]$,*

$$\theta^{i,+}(z) \geq \theta^{i+1,+}(z) \text{ and } \theta^{i,-}(z) \leq \theta^{i+1,-}(z)$$

hold.

Proof We give the proof of only $\theta^{i,+}(z) \geq \theta^{i+1,+}(z)$ because the other case can be proved in a similar way. By Eq. (5.5), for any $j \in [i + 2..n]$, we have

⁴ The reason why we use the term ‘‘pseudo-intersection’’ is that the two functions $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$ are not continuous, in general, whereas the ‘‘intersection’’ is usually defined for continuous functions.

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

Because $C_{i+1,j} - C_{i,j} = \min\{c_h \mid h \in [i+1..j-1]\} - \min\{c_h \mid h \in [i..j-1]\} \geq 0$, $\theta^{i,+j}(z) - \theta^{i+1,+j}(z) \geq 0$ holds. Therefore, we have $\theta^{i,+}(z) \geq \theta^{i+1,+}(z)$ since $\theta^{i,+}(z) = \max\{\theta^{i,+j}(z) \mid j \in [i+1..n]\}$ by Eq. (5.4). \square

Lemma 5.11 ([16]) *For any $i, j \in [1..n]$ with $i < j$,*

$$\alpha^{i,j} \leq \alpha^{i+1,j} \leq \alpha^{i+1,j+1} \text{ and } \alpha^{i,j} \leq \alpha^{i,j+1} \leq \alpha^{i+1,j+1}$$

hold.

Proof We give the proof of only $\alpha^{i,j} \leq \alpha^{i+1,j}$ because the other cases can be proved in a similar way. For any $i, j \in [1..n]$ with $i < j$ and positive constant ϵ , we have

$$\theta^{i+1,+}(\alpha^{i,j} - \epsilon) \leq \theta^{i,+}(\alpha^{i,j} - \epsilon) < \theta^{j,-}(\alpha^{i,j} - \epsilon)$$

because $\theta^{i,+}(z) \geq \theta^{i+1,+}(z)$ holds by Lemma 5.10 and $\theta^{j,-}(z)$ is a non-increasing function. This implies that $\alpha^{i,j} \leq \alpha^{i+1,j}$ holds, which completes the proof. \square

Let us show that the function w' defined in Eq. (5.16) satisfies the concave Monge property under the non-confluent flow model.

Lemma 5.12 ([16]) *The weight function w' defined in Eq. (5.16) satisfies the concave Monge property under the non-confluent flow model.*

Proof If we show that, for any $i, j \in [0..n]$ with $i < j$,

$$w'(i, j) + w'(i+1, j+1) \leq w'(i, j+1) + w'(i+1, j) \quad (5.22)$$

holds, thus completing the proof. Note that condition (5.22) holds for $i = 0$ and $j = n$, because the right-hand side of (5.22) contains $w'(0, n+1) = \infty$ and other terms are finite. Let us consider the following three cases: (1) $0 < i < j < n$, (2) $i = 0$ and $0 < j < n$, (3) $0 < i < n$ and $j = n$.

Case 1. Consider the case of $0 < i < j < n$. By Eq. (5.16), for any $(i', j') \in \{(i, j), (i, j+1), (i+1, j), (i+1, j+1)\}$, we have $w'(i', j') = \text{AT}_{\text{OPT}}(i', j')$. Recall that $\alpha^{i,j}$ is the pseudo-intersection point of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$, and we have

$$w'(i, j) = \text{AT}_{\text{OPT}}(i, j) = \phi^{i,j}(\alpha^{i,j}) = \int_0^{\alpha^{i,j}} \theta^{i,+}(z) dz + \int_{\alpha^{i,j}}^{W_n} \theta^{j,-}(z) dz. \quad (5.23)$$

For any $i, j \in [1..n-1]$ with $i < j$, Eq. (5.23) and Lemma 5.11 state that

$$\begin{aligned}
& w'(i, j+1) + w'(i+1, j) - w'(i, j) - w'(i+1, j+1) \\
&= \phi^{i,j+1}(\alpha^{i,j+1}) + \phi^{i+1,j}(\alpha^{i+1,j}) - \phi^{i,j}(\alpha^{i,j}) - \phi^{i+1,j+1}(\alpha^{i+1,j+1}) \\
&= \int_{\alpha^{i,j}}^{\alpha^{i,j+1}} \theta^{i,+}(z) dz + \int_{\alpha^{i,j+1}}^{\alpha^{i+1,j+1}} \theta^{j+1,-}(z) dz \\
&\quad - \int_{\alpha^{i,j}}^{\alpha^{i+1,j}} \theta^{j,-}(z) dz - \int_{\alpha^{i+1,j}}^{\alpha^{i+1,j+1}} \theta^{i+1,+}(z) dz. \tag{5.24}
\end{aligned}$$

Now, we show that for any $z \in [\alpha^{i,j}, \alpha^{i+1,j+1}]$,

$$\min\{\theta^{i,+}(z), \theta^{j+1,-}(z)\} \geq \max\{\theta^{j,-}(z), \theta^{i+1,+}(z)\}$$

holds. First, for any $z \in [0, W_n]$, $\theta^{i,+}(z) \geq \theta^{i+1,+}(z)$ and $\theta^{j,-}(z) \leq \theta^{j+1,-}(z)$ hold by Lemma 5.10. For any $z \geq \alpha^{i,j}$, $\theta^{i,+}(z) \geq \theta^{j,-}(z)$ holds because $\alpha^{i,j}$ is the pseudo-intersection point of $\theta^{i,+}(z)$ and $\theta^{j,-}(z)$. Similarly, for any $z < \alpha^{i+1,j+1}$, we have $\theta^{i+1,+}(z) \leq \theta^{j+1,-}(z)$. Therefore, for any $z \in [\alpha^{i,j}, \alpha^{i+1,j+1}]$, $\min\{\theta^{i,+}(z), \theta^{j+1,-}(z)\} \geq \max\{\theta^{j,-}(z), \theta^{i+1,+}(z)\}$ holds.

Thus, Eq. (5.24) continues as

$$\begin{aligned}
& w'(i, j+1) + w'(i+1, j) - w'(i, j) - w'(i+1, j+1) \\
&\geq \int_{\alpha^{i,j}}^{\alpha^{i+1,j+1}} \min\{\theta^{i,+}(z), \theta^{j+1,-}(z)\} dz - \int_{\alpha^{i,j}}^{\alpha^{i+1,j+1}} \max\{\theta^{j,-}(z), \theta^{i+1,+}(z)\} dz \\
&\geq 0,
\end{aligned}$$

and then condition (5.22) holds for any i, j with $0 < i < j < n$.

Case 2. Consider the case of $i = 0$ and $j \in [1..n-1]$. Recall that $w'(0, j) = \phi^{j,-}(0)$ and $w'(0, j+1) = \phi^{j+1,-}(0)$ by Eq. (5.16). In this case, we have

$$\begin{aligned}
& w'(0, j+1) + w'(1, j) - w'(0, j) - w'(1, j+1) \\
&= \phi^{j+1,-}(0) + \phi^{1,j}(\alpha^{1,j}) - \phi^{j,-}(0) - \phi^{1,j+1}(\alpha^{1,j+1}) \\
&= \int_0^{W_n} \theta^{j+1,-}(z) dz + \int_0^{\alpha^{1,j}} \theta^{1,+}(z) dz + \int_{\alpha^{1,j}}^{W_n} \theta^{j,-}(z) dz \\
&\quad - \int_0^{W_n} \theta^{j,-}(z) dz - \int_0^{\alpha^{1,j+1}} \theta^{1,+}(z) dz - \int_{\alpha^{1,j+1}}^{W_n} \theta^{j+1,-}(z) dz \\
&= \int_0^{\alpha^{1,j+1}} \theta^{j+1,-}(z) dz - \int_0^{\alpha^{1,j}} \theta^{j,-}(z) dz - \int_{\alpha^{1,j}}^{\alpha^{1,j+1}} \theta^{1,+}(z) dz,
\end{aligned}$$

where the last equality uses $\alpha^{1,j} \leq \alpha^{1,j+1}$ by Lemma 5.11. By Lemma 5.10, we have $\theta^{j+1,-}(z) \geq \theta^{j,-}(z)$ for any $z \in [0, W_n]$. Using the same argument as in the previous case, for any $z < \alpha^{1,j+1}$, we have $\theta^{1,+}(z) < \theta^{j+1,-}(z)$. Thus, we have

$$\begin{aligned}
& w'(0, j+1) + w'(1, j) - w'(0, j) - w'(1, j+1) \\
&= \int_0^{\alpha^{1,j+1}} \theta^{j+1,-}(z) dz - \int_0^{\alpha^{1,j}} \theta^{j,-}(z) dz - \int_{\alpha^{1,j}}^{\alpha^{1,j+1}} \theta^{1,+}(z) dz \\
&= \int_0^{\alpha^{1,j}} \{\theta^{j+1,-}(z) - \theta^{j,-}(z)\} dz + \int_{\alpha^{1,j}}^{\alpha^{1,j+1}} \{\theta^{j+1,-}(z) - \theta^{1,+}(z)\} dz \geq 0.
\end{aligned}$$

Case 3. Consider the case of $j = n$ and $i \in [1..n-1]$. Recall that $w'(i, n+1) = \phi^{i,+}(W_n)$ and $w'(i+1, n+1) = \phi^{i+1,+}(W_n)$ by Eq. (5.16). Similar to the second case, we use the facts that $\alpha^{i,n} \leq \alpha^{i+1,n}$ by Lemma 5.11, $\theta^{i,+}(z) \geq \theta^{i+1,+}(z)$ for any $z \in [0, W_n]$ by Lemma 5.10, and $\theta^{i,+}(z) \geq \theta^{n,-}(z)$ for any $z \geq \alpha^{i,n}$. Then, we have

$$\begin{aligned}
& w'(i, n+1) + w'(i+1, n) - w'(i, n) - w'(i+1, n+1) \\
&= \phi^{i,+}(W_n) + \phi^{i+1,n}(\alpha^{i+1,n}) - \phi^{i,n}(\alpha^{i,n}) - \phi^{i+1,+}(W_n) \\
&= \int_0^{W_n} \theta^{i,+}(z) dz + \int_0^{\alpha^{i+1,n}} \theta^{i+1,+}(z) dz + \int_{\alpha^{i+1,n}}^{W_n} \theta^{n,-}(z) dz \\
&\quad - \int_0^{\alpha^{i,n}} \theta^{i,+}(z) dz - \int_{\alpha^{i,n}}^{W_n} \theta^{n,-}(z) dz - \int_0^{W_n} \theta^{i+1,+}(z) dz \\
&= \int_{\alpha^{i,n}}^{W_n} \theta^{i,+}(z) dz - \int_{\alpha^{i,n}}^{\alpha^{i+1,n}} \theta^{n,-}(z) dz - \int_{\alpha^{i+1,n}}^{W_n} \theta^{i+1,+}(z) dz \\
&= \int_{\alpha^{i,n}}^{\alpha^{i+1,n}} \{\theta^{i,+}(z) - \theta^{n,-}(z)\} dz + \int_{\alpha^{i+1,n}}^{W_n} \{\theta^{i,+}(z) - \theta^{i+1,+}(z)\} dz \geq 0.
\end{aligned}$$

Thus, for any $i, j \in [0..n]$ with $i < j$, condition (5.22) holds. This implies that the function w' satisfies the concave Monge condition. \square

Acknowledgements We thank Robert Benkoczi, Binay Bhattacharya, Mordecai J. Golin, and Tsunehiko Kameda for many helpful discussions on this topic. This work was supported by JST CREST Grant Number JPMJCR1402, JSPS KAKENHI Grant Number 19H04068, and JSPS KAKENHI Grant Number 20K19746.

References

1. S. Alstrup, C. Gavoille, H. Kaplan, T. Rauhe, Nearest common ancestors: a survey and a new distributed algorithm, in *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures* (2002), pp. 258–264
2. Nadine Baumann, Martin Skutella, Earliest arrival flows with multiple sources. *Math. Operat. Res.* **34**(2), 499–512 (2009)
3. R. Belmonte, Y. Higashikawa, N. Katoh, Y. Okamoto, Polynomial-time approximability of the k -sink location problem. *CoRR* (2015). [arXiv:abs/1503.02835](https://arxiv.org/abs/1503.02835)
4. M.A. Bender, M. Farach-Colton, The lca problem revisited, in *Latin American Symposium on Theoretical Informatics* (Springer, Berlin, 2000), pp. 88–94

5. R. Benkoczi, B. Bhattacharya, Y. Higashikawa, T. Kameda, N. Katoh, Minsum k -sink problem on dynamic flow path networks, in *International Workshop on Combinatorial Algorithms*, (Springer, Berlin, 2018), pp. 78–89
6. Robert Benkoczi, Binay Bhattacharya, Yuya Higashikawa, Tsunehiko Kameda, Naoki Katoh, Minsum k -sink problem on path networks. *Theor. Comput. Sci.* **806**, 388–401 (2020)
7. B. Bhattacharya, M.J. Golin, Y. Higashikawa, T. Kameda, N. Katoh, Improved algorithms for computing k -sink on dynamic flow path networks, in *Workshop on Algorithms and Data Structures* (Springer, Berlin, 2017), pp. 133–144
8. D. Chen, M. Golin, 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. D. Chen, M.J. Golin, Minmax centered k -partitioning of trees and applications to sink evacuation with dynamic confluent flows. *CoRR* (2018). [arXiv:abs/1803.09289](https://arxiv.org/abs/1803.09289)
10. L.R. Jr Ford, D.R. Fulkerson, Constructing maximal dynamic flows from static flows. *Operations research*, **6**(3), 419–433 (1958)
11. G.N. Frederickson, D.B. Johnson, Finding k th paths and p -centers by generating and searching good data structures. *J. Algorithms* **4**(1):61–80 (1983)
12. S.L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph. *Operat. Res.* **12**(3), 450–459 (1964)
13. Y. Higashikawa, Studies on the space exploration and the sink location under incomplete information towards applications to evacuation planning (2014)
14. Y. Higashikawa, M.J. Golin, N. Katoh, Minimax regret sink location problem in dynamic tree networks with uniform capacity. *J. Graph Algorithms Appl.* **18.4**, 539–555 (2014)
15. Y. Higashikawa, M.J. Golin, N. Katoh, Multiple sink location problems in dynamic path networks. *Theor. Comput. Sci.* **607**, 2–15D (2015)
16. Y. Higashikawa, N. Katoh, J. Teruyama, K. Watase, Almost linear time algorithms for minsum k -sink problems on dynamic flow path networks, in *International Conference on Combinatorial Optimization and Applications* (Springer, Berlin, 2020), pp. 198–213
17. Bruce Hoppe, Éva. Tardos, The quickest transshipment problem. *Math. Operat. Res.* **25**(1), 36–62 (2000)
18. Satoko Mamada, Takeaki Uno, Kazuhisa Makino, Satoru Fujishige, An $o(n \log^2 n)$ algorithm for the optimal sink location problem in dynamic tree networks. *Discrete Appl. Math.* **154**(16), 2387–2401 (2006)
19. Baruch Schieber, Computing a minimum weight k -link path in graphs with the concave monge property. *J. Algorithms* **29**(2), 204–222 (1998)
20. M. Skutella, An introduction to network flows over time, in *Research Trends in Combinatorial Optimization* (Springer, Berlin, 2009), pp. 451–482

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

