The Review of Socionetwork Strategies (2019) 13:163–208
https://doi.org/10.1007/s12626-019-00047-z

ARTICLE

A Survey on Facility Location Problems in Dynamic Flow Networks

Yuya Higashikawa[1] · **Naoki Katoh**[1]

Received: 8 January 2019 / Accepted: 17 July 2019 / Published online: 4 September 2019

Abstract
This paper surveys the facility location problems in dynamic flow networks that have been actively studied in recent years. These problems have been motivated by evacuation planning which has become increasingly important in Japan. The evacuation planning problem is formulated using a dynamic flow network consisting of a graph in which a capacity as well as a transit time is associated with each edge. The goal of the problem is to find a way to send evacuees originally existing at vertices to facilities (evacuation centers) as quickly as possible. The problem can be viewed as a generalization of the classical k-center and k-median problems. In this paper we show recent results about the difficulty and approximability of a single facility location for general networks and polynomial time algorithms for k-facility location problems in path and tree networks. We also mention the minimax regret version of these problems.

Keywords Evacuation planning · Facility location · Dynamic network flow · Algorithm

1 Introduction

It has become increasingly important to establish effective evacuation planning systems against large-scale disasters, e.g., earthquakes, tsunamis, and hurricanes, for example. For this purpose, several theoretical issues have to be considered. In particular, we focus on where evacuation buildings should be located in cities. Since the Tohoku-Pacific Ocean Earthquake occurred in Japan on March 11, 2011,

✉ Yuya Higashikawa
higashikawa@sis.u-hyogo.ac.jp

Naoki Katoh
naoki.katoh@gmail.com

[1] School of Social Information Science, University of Hyogo, 8-2-1 Gakuennishi-machi, Nishi-ku, Kobe, Hyogo 651-2197, Japan

construction of tsunami evacuation buildings in large Japanese cities near the coast has become an urgent issue.

Dynamic Network To formulate the evacuation planning, we model a road network of a city as a graph in which a vertex represents an intersection associated with the number of habitants (evacuees) living close to the intersection and an edge represents a road that connects intersections associated with a transit time and a capacity which limits the flow rate entering the edge. The capacity of an edge is determined by the width and the speed limit of the corresponding road. In the graph, we are given a set of evacuation buildings (safe places) to which people evacuate when a large disaster occurs. Such a graph is called a dynamic flow network which was first introduced by Ford and Fulkerson [23]. From the viewpoint of evacuation planning, all evacuees must be sent to evacuation centers as quickly as possible. As will be mentioned later in this section, the concept of dynamic flow was first introduced to model the quickest transshipment problem: given a set of sources with available supplies and a set of sinks with required demands, the problem asks whether it is possible to send supplies to sinks in given time T to satisfy all the demands of the sinks. A sink is the point where a facility is located. This problem is very similar to the evacuation problem except for the existence of demands, and thus all the results of the quickest transshipment problems can be applied to evacuation problems. Therefore, in this paper, the number of evacuees existing at a vertex is called a supply. Notice that unlike in static flow networks, the time required to move supply from one vertex to another can be increased due to congestion caused by the capacity constraints, which requires supplies to wait at a vertex until supplies preceding it leave. Given a dynamic flow network, we are asked to transport supplies to sinks as quickly as possible.

Recently, to decrease the loss of human lives, many cities along the coast of the Pacific Ocean are planning to increase tsunami evacuation buildings. This motivates us to study the facility location problems in dynamic flow networks.

Integral Flow vs Fractional Flow Dynamic networks can be considered in integral and fractional flow models. In the integral flow model, each input value is given as an integer and each supply is regarded as a set of substantial units. In other words, a supply can be regarded as a set of evacuees, and then the edge capacity is defined as the maximum number of evacuees who can enter an edge per unit time. On the other hand, in the fractional flow model, each input value is given as a real number. Then each supply can be regarded as fluid, and edge capacity is defined as the maximum amount of supply which can enter an edge per unit time.

Optimal Facility Location Problem In this survey, we consider the facility location problems in dynamic flow networks. The problem is called the optimal facility location problem, in particular, if it requires finding a location of facilities in a given network so that all supplies are sent to the facilities as quickly as possible. For the optimality of location, the following two criteria can be naturally considered: the minimization of maximum cost and total cost (in a facility location in static flow networks, these criteria correspond to the center problem and the median problem, respectively). We explain these criteria in the integral flow model. Letting $\boldsymbol{x}$ denote a location of facilities in a dynamic flow network, assume that every evacuee goes to one of the facilities of $\boldsymbol{x}$. Then, an evacuee follows a path from a vertex where

he/she has been originally located to a facility, and each vertex v on the path stores two values which represent the time he/she arrives at v and the time he/she leaves v, respectively (assume that arriving time at the origin vertex is zero and leaving time at the facility is infinity). We call such a path with arriving times and leaving times an evacuation path.

Given a location of facilities $\boldsymbol{x}$, an evacuation to $\boldsymbol{x}$ is defined as a set of evacuation paths to $\boldsymbol{x}$ for all evacuees. An evacuation to $\boldsymbol{x}$ is called feasible if it satisfies the capacity constraint which limits the number of evacuees entering each edge per unit time. Given a location of facilities $\boldsymbol{x}$ and a feasible evacuation to $\boldsymbol{x}$, say $\mathscr{E}$, the cost of $(\boldsymbol{x}, \mathscr{E})$ for an evacuee is defined as the time required to send him/her to a facility of $\boldsymbol{x}$ along an evacuation path determined by $\mathscr{E}$. Then $max(\boldsymbol{x}, \mathscr{E})$ and $sum(\boldsymbol{x}, \mathscr{E})$ are defined as the maximum of the cost of $(\boldsymbol{x}, \mathscr{E})$ for all evacuees and the sum of the cost of $(\boldsymbol{x}, \mathscr{E})$ for all evacuees, respectively. Note that these correspond to the completion time and the average time (times the number of evacuees) by $\mathscr{E}$. The maximum cost of $\boldsymbol{x}$ and the total cost of $\boldsymbol{x}$ are defined as $\boldsymbol{\Theta}(\boldsymbol{x}) = \min_{\mathscr{E}\in\mathfrak{E}_x} max(\boldsymbol{x}, \mathscr{E})$ and $\boldsymbol{\Phi}(\boldsymbol{x}) = \min_{\mathscr{E}\in\mathfrak{E}_x} sum(\boldsymbol{x}, \mathscr{E})$, respectively, where $\mathfrak{E}_x$ is the set of all feasible evacuations to $\boldsymbol{x}$. In the fractional flow model, we define the unit as an infinitesimally small portion of supply, and the cost is defined on each infinitesimal unit. Then two criteria are defined in the same way as in the integral flow model. The minimax facility location problem and the minisum facility location problem are defined as problems that require finding a location of facilities $\boldsymbol{x}$ that minimizes $\boldsymbol{\Theta}(\boldsymbol{x})$ and $\boldsymbol{\Phi}(\boldsymbol{x})$, respectively (Tables 1, 2).

At this point, we need to mention the relationship with well-known classical facility location problems (see [20]): k-center and k-median problems which, respectively, try to find a location of k facilities that minimizes the maximum (resp. the sum) of the distance from each user located at a vertex to the nearest facility. Facility location problems treated in this paper coincide with k-center and k-median problems if the number of facilities is fixed to k, a supply corresponds to a user, and the edge capacity is infinite (i.e., no congestion occurs). In this sense, our problems generalize the conventional facility location problems.

Table 1 Table for the minimax problems

	General capacities		Uniform capacity	
	1-Facility	k-Facility	1-Facility	k-Facility
Path	$O(n \log n)$ [28, 32, 34]	$O(n \log n + k^2 \log^4 n)$ and $O(n \log^3 n)$ [8] (Theorem 9)	Trivially $O(n)$	$O(n + k^2 \log^2 n)$ and $O(n \log n)$ [8] (Theorem 10)
Tree	$O(n \log^2 n)$ [41] (Theorem 11)	$O(\max\{k, \log n\} \cdot kn \log^4 n)$ [15]	$O(n \log n)$ [32, 34] (Theorem 12)	$O(\max\{k, \log n\} \cdot kn \log^3 n)$ [15] (Theorem 13)
General graph	FPTAS [5] (Theorem 4)	FPTAS for a fixed k [5] (Theorem 4)	FPTAS [5] (Theorem 4)	FPTAS for a fixed k [5] (Theorem 4)

n, the number of vertices in a given network; k, the number of located sinks

Table 2 Table for the minisum problems

	General capacities		Uniform capacity	
	1-Facility	*k*-Facility	1-Facility	*k*-Facility
Path	Trivially $O(n^2)$	$O(kn \log^4 n)$ [7] (Theorem 16)	$O(n)$ [35] (Theorem 14)	$O(kn \log^3 n)$ [7] (Theorem 16)
Tree	Open	Open	Open	Open
General graph	Open	Open	Open	Open

n, the number of vertices in a given network; k, the number of located sinks

Here a number of questions are raised, for example: (1) What is the essential difficulty of our problems that discriminates the classical problems? (2) What is the time complexity of our problems? Although we do not have an answer to the first question (1) currently, we have a fully polynomial time approximation scheme (FPTAS) for the problem of finding an optimal location of a single facility in general graphs under the maximum cost criterion. This is contrasted with the existence of a polynomial time algorithm for the 1-center problem in general graphs. For the second question (2), regarding classical *k*-median and *k*-center problems, there exist several approximation algorithms [16, 21, 36]. For instance, there exists a 2-approximation algorithm for *k*-center problems in metric space [21, 36] while no such algorithm is known for the minimax facility location problem in dynamic flow networks.

Minimax Regret Facility Location Problem Although the above criteria are reasonable for the facility location, such criteria may not be practical since the number of evacuees in an area may vary depending on the time (e.g., in an office area in a big city, there are many people during the daytime on weekdays while there are much less people on weekends or during the night time). Therefore, to take into account the uncertainty of population distribution, we consider the maximum regret for a facility location as another evaluation criterion assuming that for each vertex, we only know an interval of vertex supply. A particular assignment of supply to each vertex is called a scenario. Here, for a location of facilities $\boldsymbol{x}$ and a scenario s, let $\boldsymbol{\Theta}^s(\boldsymbol{x})$ and $\boldsymbol{\Phi}^s(\boldsymbol{x})$ denote the maximum cost of $\boldsymbol{x}$ under s and the total cost of $\boldsymbol{x}$ under s, respectively. Also let $\boldsymbol{p}^s$ and $\boldsymbol{q}^s$ denote the minimax facility location under

Table 3 Table for the minimax regret problems adopting the maximum cost criterion

	General capacities		Uniform capacity	
	1-Facility	*k*-Facility	1-Facility	*k*-Facility
Path	Open	Open	$O(n)$ [10] (Theorem 17)	$O(kn^3)$ [28] (Theorem 18)
Tree	Open	Open	$O(n \log n)$ [10] (Theorem 19)	$O(\max\{k^2, \log^2 n\} \cdot k^2 n^2 \log^5 n)$ [26] (Theorem 20)
General graph	Open	Open	Open	Open

n, the number of vertices in a given network; k, the number of located sinks

Table 4 Table for the minimax regret problems adopting the total cost criterion

	General capacities		Uniform capacity	
	1-Facility	*k*-Facility	1-Facility	*k*-Facility
Path	Open	Open	$O(n^2 \log^2 n)$ [9] (Theorem 21)	Open
Tree	Open	Open	Open	Open
General graph	Open	Open	Open	Open

n, the number of vertices in a given network; k, the number of located sinks

s and the minisum facility location under *s*, respectively. Then, the minimax regret facility location problem is formulated as follows. The problem can be understood as a 2-person Stackelberg game, that is, the first player picks a facility location $\boldsymbol{x}$ and the second player chooses a scenario *s* that maximizes the regret defined as $\Theta^s(\boldsymbol{x}) - \Theta^s(\boldsymbol{p}^s)$ or $\Phi^s(\boldsymbol{x}) - \Phi^s(\boldsymbol{q}^s)$. The objective of the first player is to choose $\boldsymbol{x}$ that minimizes the maximum regret (Tables 3, 4).

1.1 Related Work

The quickest transshipment problem is defined by a dynamic flow with sets of sources and sinks: each source has a fixed supply, and each sink has a fixed demand. The problem is to send exactly the right amount of supply from each source to each sink with minimum overall time. This problem has been studied for over fifty years [23]. The standard technique to solve this problem is to consider discrete time steps and make a copy of the original network for every time unit from time zero to the time horizon *T*. This process produces a time-expanded network (see Sects. 2.1 and 2.2 for details). Then applying a conventional max-flow algorithm, we can determine whether the optimal value is at most *T* or not. Therefore, finding an optimal solution requires a binary search for *T*. Since the size of the time expanded network is proportional to the size of the original network times *T*, the running time of this algorithm is polynomial in the size of the original network times *T*, and thus is pseudo-polynomial.

Here it is appropriate to mention the difference between the conventional quickest transshipment problem and the evacuation problem. Since a sink stands for an evacuation center in evacuation problems, it does not actually have a demand. Instead, it has an upper bound of the number of evacuees that the center can accommodate. Therefore, the evacuation problem tries to send all supplies to sinks as quickly as possible under the capacity constraints of edges and sinks. However, the algorithms developed for the quickest transshipment problem can be applied to the evacuation problem without major modifications.

Special cases of the quickest transshipment problem have been studied. The case with a single source and a single sink has been studied by Ford and Fulkerson [24] where the maximum dynamic flow problem which sends as much flow as possible from the source to the sink within a given time bound *T* was studied. They showed

that this problem can be solved in polynomial time via one minimum cost flow computation. The quickest transshipment problem for the single source and the single sink can be reduced to the maximum dynamic flow problem by binary search. Burkard et al. [13] gave a more efficient, strongly polynomial time algorithm for this problem.

Efficient algorithms for subclasses of networks were developed [27, 38, 39, 41].

1.2 Organization

In the next section, we give definitions of dynamic flow and the time-expanded network.

In Sect. 3, we present recent results concerning optimal facility location problems in dynamic flow networks. Starting with the approximability result of k-facility location problems for general networks, we then explain efficient algorithms developed for the cases of path and tree dynamic flow networks.

In Sect. 4, we deal with the minimax regret facility location problem. We shall survey efficient algorithms recently developed for this problem with path and tree dynamic flow networks.

Section 5 concludes this paper.

2 Preliminaries

In this section, we give formal definitions of dynamic flow and a time-expanded network. Therefore, let us first explain briefly how a flow is sent on a dynamic flow network in the integral flow model.

A static flow problem is defined on a graph $G = (V, E)$ (which is assumed to be directed). Here V is the set of vertices and E is the set of edges. Each edge e is associated with a capacity $c(e)$ (which takes a positives value). The capacity $c(e)$ gives us the upper bound on the amount of the resource that can be transmitted through e. Given a source vertex s and a sink vertex t, the maximum flow problem tries to find a way that transports the resource as much as possible from s to t under capacity constraints.

As in a static flow problem, a dynamic flow problem is defined on a graph $G = (V, E)$ where each edge $e = (u, v)$ is associated with a positive capacity $c(e)$. In addition to the capacity, edge e is associated with a transit time $\tau(e)$ which implies that a flow starting from u at time t_0 arrives at v at time $t_0 + \tau(e)$. The capacity $c(e)$ limits the rate of the amount of flow that enters e per unit time. In terms of evacuation, the capacity represents the maximum possible number of evacuees that can enter the edge per unit time due to the width of the road.

Suppose that at time 0, w evacuees are now starting to traverse edge $e = (u, v)$ from u to v. Since at most $c(e)$ evacuees can enter e per unit time, $\lceil w/c(e) \rceil$ groups enter e where each group consists of $c(e)$ people except possibly the last group. The first group arrives at v at time $\tau(e)$ while the last group arrives at time $\tau(e) + \lceil w/c(e) \rceil - 1$.

Now suppose w_1 evacuees move from v_1 to v_3 through edges $e_1 = (v_1, v_2)$ and $e_2 = (v_2, v_3)$, and that there exist w_2 evacuees at v_2 who start to move towards v_3 at the same time. If some evacuees remain at v_2 when evacuees from v_1 arrive at v_2, those who came from v_1 are forced to wait at v_2. This means that a dynamic flow model has the ability to express a congestion.

2.1 Dynamic Flow

The general quickest transshipment problem will be formally defined as follows. We denote by $\mathbb{R}_+$ and $\mathbb{Z}_+$ the sets of nonnegative reals and nonnegative integers, and let $\delta_G(X)$ (resp. $\rho_G(X)$) be the set of edges $(x, y) \in E$ with $x \in X$ and $y \notin X$ (resp. $x \notin X$ and $y \in X$). Let $\mathcal{N} = (G = (V, E), S^+, S^-, w, c, \tau)$ be a dynamic flow network which consists of a directed graph $G = (V, E)$, disjoint subsets of V, S^+ and S^-, a supply function $w : V \to \mathbb{R}$ which associates each vertex $u \in S^+$ or $v \in S^-$ with a positive amount of supply (the number of evacuees) or a negative amount of supply, respectively, a capacity function $c : E \to \mathbb{R}_+ \backslash \{0\}$ which associates each edge $e \in E$ with a positive capacity, and a transit time function $\tau : E \to \mathbb{Z}_+$ which represents the time required to traverse the unit distance on any edge. S^- stands for a set of sinks which represents a set of destinations where evacuation centers exist. An illustrative example of a dynamic flow is given in Fig. 1.

The problem tries to find a dynamic flow (a way of evacuation) that minimizes the time by which all supplies have arrived at one of the sinks. A dynamic flow is defined as a function f: $E \times \mathbb{Z}_+ \to \mathbb{R}_+$ as follows. For each $e \in E$ and $\theta \in \mathbb{Z}_+$, $f(e, \theta)$ denotes the flow rate entering e at time step θ. We call f feasible if it satisfies the capacity constraint

$$0 \le f(e, \theta) \le c(e) \quad (\forall e \in E, \forall \theta \in \mathbb{Z}_+), \tag{1}$$

the flow conservation

$$\sum_{e \in \delta_G(x)} \sum_{\theta=0}^{T} f(e, \theta) \le \sum_{e \in \rho_G(x)} \sum_{\theta=0}^{T-\tau(e)} f(e, \theta) + w(x) \quad (\forall x \in V, \forall T \in \mathbb{Z}_+), \tag{2}$$

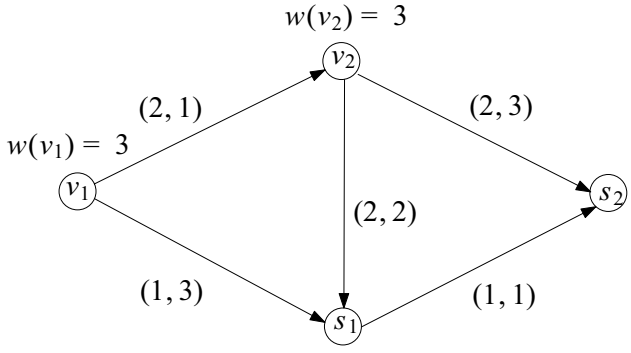

Fig. 1 Example of a dynamic flow network. The pair of numbers $(\cdot, \cdot)$ attached to each edge e represents $(c(e), \tau(e))$

and the demand constraint

$$\sum_{e\in\rho_G(S^-)}\sum_{\theta=0}^{T-\tau(e)} f(e,\theta) = \sum_{v\in V} w(v) \quad (\exists T \in \mathbb{Z}_+)\,. \tag{3}$$

Notice that in (2) we allow storage at intermediate vertices. For a feasible dynamic flow f, we define the evacuation time of f by the minimum time step T satisfying (3). Then, the evacuation problem tries to find the minimum evacuation time among all feasible dynamic flows as well as an optimal dynamic flow which attains the minimum evacuation time.

2.2 Time-Expanded Networks

To find an optimal dynamic flow for the evacuation problem, Ford and Fulkerson [23, 24] introduced the time-expanded network $\mathcal{N}(T)$ which is a static flow network for a dynamic flow network $\mathcal{N}$ with a time horizon T (see Fig. 2). The vertex set of $\mathcal{N}(T)$ consists of two parts defined as follows. The first part contains a vertex $x(\theta)$ for each $x \in V$ and $\theta \in \{0,\dots,T\}$, and the second part contains a vertex x^* for each $x \in V \setminus S^-$. On the other hand, the edge set of $\mathcal{N}(T)$ consists of three parts defined as follows. The first part contains an edge $e(\theta) = (x(\theta), y(\theta+\tau(e)))$ with a capacity $c(e)$ for each $e = (x,y) \in E$ and $\theta \in \{0,\dots,T-\tau(e)\}$, and the second part contains an edge $(x(\theta), x(\theta+1))$ with an infinite capacity for each $x \in V \setminus S^-$ and $\theta \in \{0,\dots,T-1\}$. The edges of the second part are called holdover edges. Finally, the third part contains an edge $(x^*, x(\theta))$ with an infinite capacity for each $x \in V \setminus S^-$ and $\theta \in \{0,\dots,T\}$. We define the source set and the sink set of $\mathcal{N}(T)$ by $\{x^* \mid x \in V \setminus S^-\}$ and $\{s(\theta) \mid \theta \in \{0,\dots,T\}\}$, respectively. For each $x \in V \setminus S^-$, the supply of x^* is set to $w(x)$. Notice that the size of the

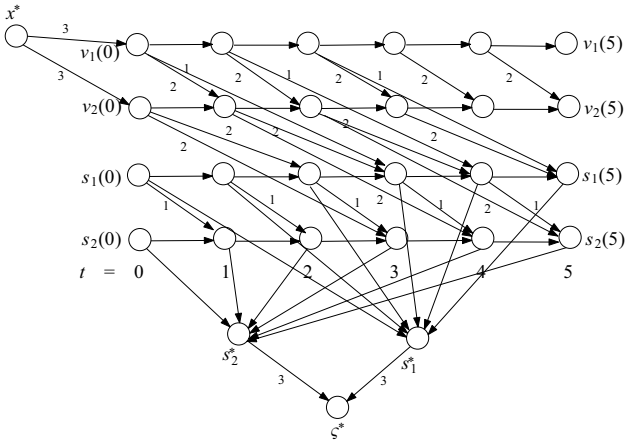

Fig. 2 A time-expanded network $\mathcal{N}(5)$ for the dynamic flow network $\mathcal{N}$ illustrated in Fig. 1. A number attached to a vertex represents its supply. A number attached to an edge represents its capacity

time-expanded network $\mathcal{N}(T)$ is proportional to T and thus is pseudo-polynomial in the input size.

For each $s \in S^-$ and for the corresponding sink set $\{s(\theta) \mid 0 \leq \theta \leq T\}$, we introduce a super sink s^* as well as an edge $(s(\theta), s^*)$ with $+\infty$ capacity. Furthermore, we introduce a general super sink ς^* as well as an edge (s^*, ς^*) with capacity $b(s)$ which denotes the capacity of an evacuation center s.

Theorem 1 (Ford and Fulkerson [23, 24]) *Given a dynamic flow network $\mathcal{N}$ and a time horizon T, there exists a feasible dynamic flow in $\mathcal{N}$ whose evacuation time is less than or equal to T if and only if the value of a maximum flow from x^* to ς^* is equal to $\sum_{v \in V} w(v)$. Furthermore, we can construct a feasible dynamic flow in $\mathcal{N}$ whose evacuation time is less than or equal to T from a maximum flow ξ in $\mathcal{N}(T)$ whose flow value is equal to $\sum_{v \in V} w(v)$ by setting $f(e, \theta) = \xi(e(\theta))$ for each $e \in E$ and $\theta \in \{0, \ldots, T - \tau(e)\}$.*

Since the size of a time-expanded network is pseudo-polynomial in input size and computing the maximum flow can be done in polynomial time in input size, this theorem implies that testing whether all evacuees can be evacuated to one of the facilities (i.e., sinks) within time T can be done in pseudo-polynomial time. By applying a binary search over T, a dynamic flow that minimizes the maximum cost can be obtained in pseudo-polynomial time.

Theorem 2 *An optimal dynamic flow under the maximum cost criterion can be obtained in pseudo-polynomial time.*

Time-expanded networks can also be used for solving a dynamic flow problem under total cost criterion. Suppose we are given a sufficiently large time horizon T. Let us consider the minimum cost flow problem $Q(T)$ defined on a time-expanded network $\mathcal{N}(T)$ where the cost of an edge $e(\theta) = (x(\theta), y(\theta + \tau(e)))$ with a capacity $c(e)$ for each $e = (x, y) \in E$ is defined to be $\tau(e)$, and the cost of a holdover edge $(x(\theta), x(\theta + 1))$ is defined to be 1 while the cost of all the other edges are defined to be zero. Then the cost of flow ξ from x^* to ς^* with the flow value equal to $\sum_{v \in V} w(v)$ is equivalent to the total cost of the corresponding dynamic flow. Since the minimum cost flow problem can be solved in polynomial time in input size (see [2]) and the size of the time-expanded network $\mathcal{N}(T)$ is pseudo-polynomial, the following theorem follows.

Theorem 3 *An optimal dynamic flow under the total cost criterion can be obtained in pseudo-polynomial time.*

Contrary to the maximum cost criterion, a polynomial time algorithm for computing an optimal dynamic flow under the total cost criterion is not known.

2.3 Dynamic Network Models in Facility Location and Quickest Transshipment

As mentioned in Sect. 2.1, a dynamic flow network consists of a directed graph in quickest transshipment problems. In contrast to this, all previous studies for facility location problems (which will be shown in the following sections) have assumed that a dynamic flow network consists of an undirected graph. We thus also assume the undirected model in the following sections. Note that under the optimal evacuation in an undirected dynamic flow network, no edge is ever used bidirectionally at the same time because if so, changing the route and/or the destination does not increase the maximum/total cost. In addition, we notice that an edge may be used in one direction at a time and in another direction at another time. This implies that an undirected dynamic flow network can be transformed to an equivalent directed one by replacing every undirected edge uv of capacity $c(uv)$ and transit time $\tau(uv)$ with two directed edges (u, v) and (v, u) such that each one has the same capacity $c(uv)$ and the same transit time $\tau(uv)$.

Also in the following sections, we treat dynamic flow networks where each edge is associated with a positive length, instead of a transit time. Additionally, we define τ as a transit time required for traversing the unit distance, i.e., if the distance between two points in a dynamic flow network is d, an evacuee can traverse the distance in time τd (with no congestion). This is because in facility location problems, facilities have been usually assumed to be located at any point in a network, so it seems to be natural that the transit time is defined as a function of the distance.

3 Optimal Facility Location Problems in Dynamic Networks

Recently, minimax facility location problems in dynamic flow networks have been studied by several researchers. First, Mamada et al. [41] studied the minimax 1-facility location problem in a dynamic flow tree network assuming that the facility must be located at a vertex, and proposed an $O(n \log^2 n)$ time algorithm. Higashikawa et al. [32, 34] also studied the same problem as [41] by assuming that the edge capacity is uniform and a facility can be located at any point in the network, and proposed an $O(n \log n)$ time algorithm. For the minimax k-facility location problems in a dynamic flow tree network, Chen and Golin [14] provided an algorithm that costs $O(k^2 n \log^5 n)$ time and $O(k^2 n \log^4 n)$ time for cases with general edge capacities and uniform edge capacity, respectively. In a recent paper [15] on arXiv, the authors of [14] have improved the time complexities to $O(\max\{k, \log n\} \cdot kn \log^4 n)$ and $O(\max\{k, \log n\} \cdot kn \log^3 n)$, respectively. Other than tree networks, Higashikawa et al. [33] treated the minimax k-facility location problem in a dynamic flow path network with uniform edge capacity, and proposed an $O(kn \log n)$ time algorithm, which was improved to $O(kn)$ by the same authors in [35]. Also, Higashikawa showed in his doctoral dissertation [28] that the minimax 1-facility and k-facility location problems in a dynamic flow path network with general edge capacities can be solved in $O(n \log n)$ time and $O(kn^2 \log n)$ time, respectively. Finally, for the minimax k-facility location problem in a dynamic flow path network with general edge capacities, Bhattacharya et al. [8] have provided an $O(n \log n + k^2 \log^4 n)$ time

algorithm and an $O(n \log^3 n)$ time algorithm. In [8], the authors have also shown that there exist an $O(n + k^2 \log^2 n)$ time algorithm and an $O(n \log n)$ time algorithm for the case with uniform edge capacity, which together improve upon the previous results for any k. For general networks, Belmonte et al. [5] showed that the minimax k-facility location problem admits a fully polynomial time approximation scheme (FPTAS) for every fixed k, and that the problem is $W[1]$-hard when parameterized by k (see [22, 42] for details of parameterized complexity).

On the other hand, the minisum facility location problems in dynamic flow networks have not been studied much except for the problem in path networks [6, 7, 33, 35]. Paper [33] treated the minisum k-facility location problem in a dynamic flow path network with uniform edge capacity, and proposed an $O(kn^2)$ time algorithm, which was improved to $\min\{O(n^2\sqrt{k \log n} + n^2 \log n), n^2 2^{O(\sqrt{\log k \log\log n})}\}$ by the same authors in [35]. Recently, Benkoczi et al. [6] have developed an $O(kn \log^3 n)$ time algorithm, which is the best so far. In [6], the authors also treated the case with general edge capacities and proposed an $O(kn^2 \log^2 n)$ time algorithm, which was improved to $O(kn \log^4 n)$ by the same authors in [7].

In the rest of this section, we define minimax and minisum facility location problems in dynamic flow networks, introduce basic ideas of algorithms to solve the problems and several properties upon which the algorithms are based.

3.1 Dynamic Network Under Fixed Supplies

A dynamic flow network under fixed supplies $\mathcal{N} = (G = (V, E), w, l, c, \tau)$ consists of an undirected graph $G = (V, E)$, function w which associates each vertex $v \in V$ with a positive supply, function l which associates each edge $e \in E$ with a positive length, function c which associates each edge $e \in E$ with a positive capacity, and a positive constant τ which represents the time required by the flow to traverse the unit distance in the network. In Sect. 3, for a vertex $v \in V$, we abuse $w(v)$ to denote the amount of supply of v although $w(v)$ represents the supply of v as a set of substantial units.

3.2 Minimax Facility Location Problems

We consider the facility location problems in a dynamic flow network with the integral flow model. Given a location of facilities $\boldsymbol{x}$ and a feasible evacuation to $\boldsymbol{x}$, say $\mathscr{E}$, the cost of $(\boldsymbol{x}, \mathscr{E})$ for an evacuee is defined as the time required to send him/her to a facility of $\boldsymbol{x}$ along an evacuation path determined by $\mathscr{E}$. Then $max(\boldsymbol{x}, \mathscr{E})$ is defined as the maximum cost of $(\boldsymbol{x}, \mathscr{E})$ for all evacuees. The maximum cost of $\boldsymbol{x}$ is defined as

$$\Theta(\boldsymbol{x}) = \min\{max(\boldsymbol{x}, \mathscr{E}) \mid \mathscr{E} \in \mathfrak{E}_x\}, \tag{4}$$

where $\mathfrak{E}_x$ is a set of all feasible evacuations to $\boldsymbol{x}$. In the fractional flow model, we define the unit as the infinitesimally small portion of supply, and the cost is defined on each infinitesimal unit. Then two criteria are defined in the same way as in the integral flow model. Here, we treat a problem that requires finding $\boldsymbol{x}$ in a dynamic flow network minimizing $\Theta(\boldsymbol{x})$.

3.2.1 *k*-Facility Location in General Graphs

Very recently, the minimax *k*-facility location problem in general dynamic flow networks has been studied by Belmonte et al. [5]. Given a dynamic flow network $\mathcal{N} = (G, w, l, c, \tau)$ where $G = (V, E)$ is a general graph, the problem requires finding a location of k facilities $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in G^k$ that minimizes $\Theta(\boldsymbol{x})$. Here, the notation G is abused to denote the set of all points on G. The authors of [5] showed that the minimax *k*-facility location problem in dynamic flow networks admits a fully polynomial time approximation scheme (FPTAS) for every fixed k, and that the problem is $W[1]$-hard when parameterized by k. Basically, they used the previous results of Hoppe et al. [37] for the quickest transshipment problem, which requires the computation of $\Theta(\boldsymbol{x})$ for a given $\boldsymbol{x}$ and finding an optimal evacuation to $\boldsymbol{x}$. Although the algorithm in [37] is for the case in which an input graph is directed, the authors of [5] showed that the algorithm can be applied to the undirected case by replacing every undirected edge uv of capacity $c(uv)$ with two directed edges (u, v) and (v, u) of capacity $c(uv)$. Then their FPTAS for the minimax *k*-facility location problem is roughly described as follows. For a positive $\epsilon > 0$, the algorithm places $1/\epsilon$ points on every edge at even intervals. Letting χ be the set of original vertices and the points so generated, the algorithm computes a location of k facilities $\boldsymbol{x}$ that minimizes $\Theta(\boldsymbol{x})$ for $\boldsymbol{x} \in \chi^k$ using the algorithm of [37].

Theorem 4 [5] *The minimax k-facility location problem in dynamic flow networks admits FPTAS for a fixed k.*

Theorem 5 [5] *The minimax k-facility location problem in dynamic flow networks is W*[1]*-hard when parameterized by k.*

For the minisum *k*-facility location problem in the general dynamic flow network, there has been no theoretical result.

3.2.2 *k*-Facility Location in Paths

The minimax *k*-facility location problems in a dynamic flow path network have been studied in [8, 28, 33, 35]. We review the algorithms proposed in [8, 28, 33, 35]. Given a dynamic flow path network $\mathcal{N} = (P = (V, E), w, l, c, \tau)$ where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_{n-1}\}$ such that v_i and v_{i+1} are endpoints of e_i for $1 \le i \le n-1$, the problem requires finding a location of k facilities $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in P^k$ that minimizes $\Theta(\boldsymbol{x})$. In the following, for integers p, i, j satisfying $1 \le p \le k$ and $1 \le i \le j \le n$, let $\Theta_{\mathrm{OPT}}(p, i, j)$ denote the cost of the minimax p-facility location in P_{ij}. Here, the notation P is abused to denote the set of all points on P, and for any point $x \in P$, the notation x is abused to denote the distance from v_1 to x. It is assumed that all units of supply originally located at a vertex are sent to the same facility. The following description applies to both the cases of uniform edge capacity and general edge capacities unless specifically mentioned. Notice that papers [33, 35] only deal with the case of uniform edge capacity while [28] extends it to the case of general edge capacities and

[8] treats both of two cases. In the following, we use the notation x to denote $\boldsymbol{x}$ when we consider the 1-facility location.

Basic properties For integers i, j satisfying $1 \le i \le j \le n$, let $P_{i,j}$ denote the subpath of P between v_i and v_j (including v_i and v_j). Suppose that a facility is located at a point $x \in P_{i,j}$. Let $\Theta_{i,j}(x)$ denote the maximum cost of x for all supplies on $P_{i,j}$, i.e., the minimum time required to send to x all supplies on $P_{i,j}$. Also let $\Theta^i_L(x)$ (resp. $\Theta^j_R(x)$) denote the minimum time required to send to x all supplies on the part of $P_{i,j}$ between v_i and x (resp. x and v_j) where $\Theta^i_L(v_i) = 0$ and $\Theta^j_R(v_j) = 0$. Then, $\Theta_{i,j}(x)$ is the maximum of $\Theta^i_L(x)$ and $\Theta^j_R(x)$, i.e.,

$$\Theta_{i,j}(x) = \max\left\{\Theta^i_L(x), \Theta^j_R(x)\right\}. \tag{5}$$

This formula was developed by [33, 35] which has also been shown in [18, 29]. Notice that this formula holds also for the case of general edge capacities. Now, assume that x is located on an edge e_{s_x} (not including endpoints) satisfying $i \le s_x \le j-1$. For the integral flow model, $\Theta^i_L(x)$ and $\Theta^j_R(x)$ are expressed as follows:

$$\Theta^i_L(x) = \max\left\{\tau(x - v_l) + \left\lceil \frac{\sum_{i\le h\le l} w(v_h)}{c'} \right\rceil - 1 \;\middle|\; i \le l \le s_x \right\}, \tag{6}$$

$$\Theta^j_R(x) = \max\left\{\tau(v_l - x) + \left\lceil \frac{\sum_{l\le h\le j} w(v_h)}{c'} \right\rceil - 1 \;\middle|\; s_x + 1 \le l \le j \right\}, \tag{7}$$

where c' is the uniform edge capacity. From these, $\Theta^i_L(x)$ and $\Theta^j_R(x)$ are expressed in the fractional flow model as follows:

$$\Theta^i_L(x) = \max\left\{\tau(x - v_l) + \frac{\sum_{i\le h\le l} w(v_h)}{c'} \;\middle|\; i \le l \le s_x \right\}, \tag{8}$$

$$\Theta^j_R(x) = \max\left\{\tau(v_l - x) + \frac{\sum_{l\le h\le j} w(v_h)}{c'} \;\middle|\; s_x + 1 \le l \le j \right\}. \tag{9}$$

Formulae (6) and (7) were first shown by Kamiyama et al. [38] while (8) and (9) were shown in [33, 35]. Note that $\Theta^i_L(x)$ (resp. $\Theta^j_R(x)$) is a piecewise linear strictly increasing (resp. decreasing) function of x. Therefore, function $\Theta_{i,j}(x)$ is unimodal in x, and there exists the unique point that minimizes $\Theta_{i,j}(x)$.

For the general edge capacities, formulae are also developed both in the integral and fractional flow models. We only give those for the fractional flow model. Assuming that x is located on an edge e_{s_x} (not including endpoints) satisfying $i \le s_x \le j-1$, $\Theta^i_L(x)$ and $\Theta^i_R(x)$ are expressed as follows:

$$\Theta^i_L(x) = \max\left\{\tau(x - v_l) + \frac{\sum_{i\le h\le l} w(v_h)}{\min_{l\le h\le s_x} c(e_h)} \;\middle|\; i \le l \le s_x \right\}, \tag{10}$$

$$\Theta_R^j(x) = \max\left\{ \tau(v_l - x) + \frac{\sum_{l\le h\le j} w(v_h)}{\min_{s_x\le h\le l-1} c(e_h)} \;\middle|\; s_x + 1 \le l \le j \right\}. \tag{11}$$

These formulae are developed by Higashikawa [28] in which the correctness proof of them is given and those for the integral flow model are also given.

Sketch of Algorithms Based on Dynamic Programming In [28, 33, 35], the basic idea to solve the problems is a dynamic programming, i.e., the algorithm repeatedly solves the p-facility location problem in $P_{i,j}$ for $1 \le p \le k$ and $1 \le i \le j \le n$. Then the following recursion holds for $p \ge 2$:

$$\Theta_{\text{OPT}}(p,i,j) = \min\left\{ \max\left\{\Theta_{\text{OPT}}(p-1,i,t), \Theta_{\text{OPT}}(1,t+1,j)\right\} \;\middle|\; i \le t \le j-1 \right\}. \tag{12}$$

To solve the problem effectively, the authors of [33, 35] proved monotonic properties shown in Lemmas 1 and 2 . Let $d_p(j)$ denote an integer t minimizing $\max\{\Theta_{\text{OPT}}(p-1,1,t), \Theta_{\text{OPT}}(1,t+1,j)\}$ for $1 \le t \le j-1$, i.e.,

$$\Theta_{\text{OPT}}(p,1,j) = \max\left\{\Theta_{\text{OPT}}(p-1,1,d_p(j)), \Theta_{\text{OPT}}(1,d_p(j)+1,j)\right\}. \tag{13}$$

Lemma 1 [33, 35] *For integers p, j satisfying $2 \le p \le k$ and $1 \le j \le n-1$, $d_p(j) \le d_p(j+1)$ holds.*

Proof To prove Lemma 1, we note the following fundamental property. □

Claim 1 *For integers p satisfying $1 \le p \le k$, and i,j,i',j' satisfying $1 \le i' \le i \le j \le j' \le n$, $\Theta_{\text{OPT}}(p,i,j) \le \Theta_{\text{OPT}}(p,i',j')$ holds.*

We prove Lemma 1 by contradiction: there exist integers p, j satisfying $2 \le p \le k$ and $1 \le j \le n-1$ such that $d_p(j) > d_p(j+1)$ holds. For ease of notation in the proof, we use the notation A, B, C, D, E and F as follows:

$$\begin{aligned} A &= \Theta_{\text{OPT}}(p-1,1,d_p(j)), & B &= \Theta_{\text{OPT}}(1,d_p(j)+1,j),\\ C &= \Theta_{\text{OPT}}(p-1,1,d_p(j+1)), & D &= \Theta_{\text{OPT}}(1,d_p(j+1)+1,j+1),\\ E &= \Theta_{\text{OPT}}(1,d_p(j+1)+1,j), & F &= \Theta_{\text{OPT}}(1,d_p(j)+1,j+1). \end{aligned} \tag{14}$$

From the assumption of $d_p(j) > d_p(j+1)$ and Claim 1, we can derive the following inequalities:

$$C \le A, \tag{15}$$

$$B \le E \le D, \tag{16}$$

$$B \le F \le D. \tag{17}$$

Since $d_p(j)$ minimizes $\max\{\Theta_{\text{OPT}}(p-1,1,h), \Theta_{\text{OPT}}(1,h+1,j)\}$ over h, we have the following inequality:

$$\max\{A, B\} \leq \max\{C, E\}. \tag{18}$$

Also, without loss of generality, we assume that $d_p(j+1)$ is maximized unless the cost increases. By this assumption, we have the following inequality:

$$\max\{C, D\} < \max\{A, F\}. \tag{19}$$

Then, we consider three cases: [Case 1] $A \leq B$; [Case 2] $D \leq C$; [Case 3] $B < A$ and $C < D$.

[Case 1]: By (15), (17) and the condition of $A \leq B$, we have $C \leq A \leq F \leq D$, which contradicts (19).

[Case 2]: By (15), (16) and the condition of $D \leq C$, we have $B \leq E \leq C \leq A$. By this and (18), we have $A = C$. Also, by (15), (17) and the condition of $D \leq C$, we have $F \leq D \leq C \leq A$. By this and (19), we have $C < A$, which contradicts $A = C$.

[Case 3]: By (18) and the condition of $B < A$, we have

$$A \leq \max\{C, E\}. \tag{20}$$

Also, by (19) and the condition of $C < D$, we have

$$D < \max\{A, F\}. \tag{21}$$

If $F \leq A$ holds, we have $D < \max\{C, E\}$ by (20) and (21), which contradicts the condition of $C < D$ or (16). If $A < F$ holds, we have $D < F$ by (21), which contradicts (17). □

Let $x^*(i,j)$ denote the minimax 1-facility location in $P_{i,j}$, i.e., $x^*(i,j)$ minimizes $\Theta_{i,j}(x)$ for $x \in P_{i,j}$.

Lemma 2 [33, 35] *For integers i, j, i', j' satisfying $1 \leq i \leq j \leq n, 1 \leq i' \leq j' \leq n, i \leq i'$ and $j \leq j'$, $x^*(i,j) \leq x^*(i',j')$ holds.*

Proof To prove Lemma 2, we confirm the following claim (refer to the definitions of (8) and (9)). □

Claim 2

(i) *For integers i, j satisfying $1 \leq i \leq j \leq n$ and points $x, y \in P$ satisfying $v_i \leq x \leq y \leq v_j$, $\Theta^i_L(x) \leq \Theta^i_L(y)$ and $\Theta^j_R(x) \geq \Theta^j_R(y)$* hold.

(ii) *For integers i, j satisfying $1 \leq i \leq j \leq n$ and points $x, y \in P$ satisfying $x \leq v_i \leq v_j \leq y$, $\Theta^i_R(x) \leq \Theta^j_R(x)$ and $\Theta^i_L(y) \geq \Theta^j_L(y)$ hold.*

We prove Lemma 2 by contradiction: there exist integers i, j, i', j' satisfying $1 \leq i \leq j \leq n$, $1 \leq i' \leq j' \leq n$, $i \leq i'$ and $j \leq j'$ such that $x^*(i,j) > x^*(i',j')$ holds. By this assumption, we have the following inequality:

$$i \leq i' \leq x^*(i',j') < x^*(i,j) \leq j \leq j'. \tag{22}$$

For ease of notation in the proof, we use the notation A, B, C, D, E, F, G and H as follows:

$$\begin{aligned} A &= \Theta^i_L(x^*(i,j)), \quad B = \Theta^j_R(x^*(i,j)),\\ C &= \Theta^{i'}_L(x^*(i',j')), \quad D = \Theta^{j'}_R(x^*(i',j')),\\ E &= \Theta^i_L(x^*(i',j')), \quad F = \Theta^j_R(x^*(i',j')),\\ G &= \Theta^{i'}_L(x^*(i,j)), \quad H = \Theta^{j'}_R(x^*(i,j)). \end{aligned} \tag{23}$$

From (22) and Claim 2, we can derive the following inequalities:

$$C \le E \le A, \tag{24}$$

$$C \le G \le A, \tag{25}$$

$$B \le F \le D, \tag{26}$$

$$B \le H \le D. \tag{27}$$

Since $x^*(i,j)$ and $x^*(i',j')$ are the unique points that minimize $\Theta_{i,j}(x) = \max\{\Theta^i_L(x), \Theta^j_R(x)\}$ and $\Theta_{i',j'}(x) = \max\{\Theta^{i'}_L(x), \Theta^{j'}_R(x)\}$, respectively (refer to (5)), we have the following inequalities:

$$\max\{A,B\} < \max\{E,F\}, \tag{28}$$

$$\max\{C,D\} < \max\{G,H\}. \tag{29}$$

Then, we consider three cases: [Case 1] $A \le B$; [Case 2] $D \le C$; [Case 3] $B < A$ and $C < D$.

[Case 1]: By (25), (27) and the condition of $A \le B$, we have $C \le G \le H \le D$, which contradicts (29).

[Case 2]: By (24), (26) and the condition of $D \le C$, we have $B \le F \le E \le A$, which contradicts (28).

[Case 3]: By (28) and the condition of $B < A$, we have

$$A < \max\{E,F\}. \tag{30}$$

Also, by (29) and the condition of $C < D$, we have

$$D < \max\{G,H\}. \tag{31}$$

If $F \le E$ holds, we have $A < E$ by (30), which contradicts (24). Also, if $G \le H$ holds, we have $D < H$ by (31), which contradicts (27). If $E < F$ and $H < G$ hold, we have $A < F \le D < G$ by (26), (30) and (31), that is, $A < G$ holds, which contradicts (25). □

We first overview an $O(kn^2 \log n)$ time algorithm by [28] for the case of general edge capacities. The algorithm computes $\Theta_{\mathrm{OPT}}(p,1,1)$, ⋯, $\Theta_{\mathrm{OPT}}(p,1,n)$ for $p = 1,2,\ldots,k$ in this order, using a dynamic programming approach based on the recursion (12). For some integers p, j satisfying $2 \le p \le k$ and $2 \le j \le n$, let us see how to obtain $\Theta_{\mathrm{OPT}}(p,1,j)$. Supposing that $\Theta_{\mathrm{OPT}}(1,1,1)$, ⋯, $\Theta_{\mathrm{OPT}}(p-1,1,n)$, $\Theta_{\mathrm{OPT}}(p,1,1)$, ⋯, $\Theta_{\mathrm{OPT}}(p,1,j-1)$ have already been obtained, we consider computing $\Theta_{\mathrm{OPT}}(p,1,j)$ based on all information obtained by then. Here, for integers p, j satisfying $2 \le p \le k$ and $2 \le j \le n$, let $f_{p,j}(t)$ denote a function of $t \in \{1,2,\ldots,j-1\}$ defined as

$$f_{p,j}(t) = \max\{\Theta_{\text{OPT}}(p-1,1,t), \Theta_{\text{OPT}}(1,t+1,j)\}. \quad (32)$$

Note that $f_{p,j}(t)$ is unimodal in t and $d_p(j)$ is an integer t that minimizes $f_{p,j}(t)$. In the following, we show how the algorithm obtains $\Theta_{\text{OPT}}(p,1,j) = f_{p,j}(d_p(j))$.

First, the algorithm computes

$$f_{p,j}(d_p(j-1)) = \max\{\Theta_{\text{OPT}}(p-1,1,d_p(j-1)), \Theta_{\text{OPT}}(1,d_p(j-1)+1,j)\}. \quad (33)$$

Since $\Theta_{\text{OPT}}(p-1,1,d_p(j-1))$ has been obtained in a previous step of dynamic programming, $f_{p,j}(d_p(j-1))$ can be obtained only by computing $\Theta_{\text{OPT}}(1,d_p(j-1)+1,j)$.

Next, since $d_p(j-1) \le d_p(j)$ holds by Lemma 1, the algorithm continues to compute $f_{p,j}(t)$ for $t = d_p(j-1)+1, d_p(j-1)+2, \ldots$ in this order. Since $f_{p,j}(t)$ is unimodal in t and $f_{p,j}(t)$ is minimized when $t = d_p(j)$, if the algorithm reaches the first integer $t^* \ge d_p(j-1)$ such that $f_{p,j}(t^*) \le f_{p,j}(t^*+1)$, it outputs t^* as $d_p(j)$. Then, the algorithm also outputs $f_{p,j}(t^*)$ as $\Theta_{\text{OPT}}(p,1,j)$. For some i ($\ge d_p(j-1)+1$), the algorithm computes

$$f_{p,j}(i) = \max\{\Theta_{\text{OPT}}(p-1,1,i), \Theta_{\text{OPT}}(1,i+1,j)\}. \quad (34)$$

Since $\Theta_{\text{OPT}}(p-1,1,i)$ has been obtained in a previous step of dynamic programming, $f_{p,j}(i)$ can be obtained only by computing $\Theta_{\text{OPT}}(1,i+1,j)$.

Let us turn to the time complexity of the above mentioned algorithm. For ease of explanation, we here assume that facilities are located on vertices although it is allowed in [28] that facilities are located at any point on a path. Then, during the computation of $\Theta_{\text{OPT}}(p,1,1), \ldots, \Theta_{\text{OPT}}(p,1,n)$ for a fixed p, an algorithm just computes $\Theta_{\text{OPT}}(1,i,j)$ while increasing i or j one by one as mentioned above, which implies that the number of such computations is $O(n)$ for a fixed p. Therefore, up to $\Theta_{\text{OPT}}(k,1,n)$, $O(kn)$ computations are required in total. Each computation of $\Theta_{\text{OPT}}(1,i,j)$ can be obtained by applying a binary search based on the unimodality of $\Theta_{i,j}(x)$, and $\Theta_{i,j}(x)$ for a fixed x can be computed in $O(n)$ time using the formulae (10) and (11), i.e., $\Theta_{\text{OPT}}(1,i,j)$ can be computed in $O(n \log n)$ time. These imply an $O(kn^2 \log n)$ time algorithm for minimax k-facility location problem in a dynamic flow path network with general edge capacities.

Theorem 6 [28] *The minimax k-facility location problem in a dynamic flow path network with general edge capacities can be solved in* $O(kn^2 \log n)$ *time.*

For the case of uniform edge capacity, an algorithm by [33, 35] basically applies an similar approach as the algorithm by [28] for the case of general edge capacities, but there are differences as follows. In the following, we call the algorithm by [28] and one by [33, 35] ALG_{GC} and ALG_{UC}, respectively.

Similarly to ALG_{GC}, when computing $\Theta_{\text{OPT}}(p,1,j)$, ALG_{UC} first computes

$$f_{p,j}(d_p(j-1)) = \max\{\Theta_{\text{OPT}}(p-1,1,d_p(j-1)), \Theta_{\text{OPT}}(1,d_p(j-1)+1,j)\},$$

which is the same as (33). Note that $d_p(j-1)$ has been obtained and $f_{p,j-1}(d_p(j-1))$ has computed as

$$f_{p,j-1}(d_p(j-1)) = \max\{\Theta_{\mathrm{OPT}}(p-1,1,d_p(j-1)), \Theta_{\mathrm{OPT}}(1,d_p(j-1)+1,j-1)\}. \tag{35}$$

To obtain $f_{p,j}(d_p(j-1))$, ALG_{GC} independently computes $\Theta_{\mathrm{OPT}}(1,d_p(j-1)+1,j)$ as mentioned above. Unlike this, ALG_{UC} updates $\Theta_{\mathrm{OPT}}(1,d_p(j-1)+1,j-1)$ (which has already been obtained in (35)) to $\Theta_{\mathrm{OPT}}(1,d_p(j-1)+1,j)$. In this updating, letting $i = d_p(j-1)+1$, ALG_{UC} computes $\Theta_{i,j}(x)$ from $x = x^*(i,j-1)$ to $x^*(i,j)$, which satisfy $x^*(i,j-1) \le x^*(i,j)$ by Lemma 2.

Also similarly to ALG_{GC}, ALG_{UC} continues to compute $f_{p,j}(t)$ for $t = d_p(j-1)+1, d_p(j-1)+2, \ldots$ in this order, and outputs t^* as $d_p(j)$ if it reaches the first integer $t^* \ge d_p(j-1)$ such that $f_{p,j}(t^*) \le f_{p,j}(t^*+1)$. For some $i\ (\ge d_p(j-1)+1)$, ALG_{UC} computes

$$f_{p,j}(i) = \max\{\Theta_{\mathrm{OPT}}(p-1,1,i), \Theta_{\mathrm{OPT}}(1,i+1,j)\},$$

which is the same as (34). Suppose that $f_{p,j}(i-1)$ has been obtained as

$$f_{p,j}(i-1) = \max\{\Theta_{\mathrm{OPT}}(p-1,1,i-1), \Theta_{\mathrm{OPT}}(1,i,j)\}. \tag{36}$$

To obtain $f_{p,j}(i)$, ALG_{GC} independently computes $\Theta_{\mathrm{OPT}}(1,i+1,j)$ as mentioned above. Unlike this, ALG_{UC} updates $\Theta_{\mathrm{OPT}}(1,i,j)$ (which has already been obtained in (36)) to $\Theta_{\mathrm{OPT}}(1,i+1,j)$. In this updating, the algorithm computes $\Theta_{i+1,j}(x)$ from $x = x^*(i,j)$ to $x^*(i+1,j)$, which satisfy $x^*(i,j) \le x^*(i+1,j)$ by Lemma 2.

Therefore, ALG_{UC} computes $\Theta_{i,j}(x)$ while increasing i or j one by one or moving x monotonically to the next vertex during the computation of $\Theta_{\mathrm{OPT}}(p,1,1)$, $\ldots, \Theta_{\mathrm{OPT}}(p,1,n)$ for a fixed p (we also assume that facilities are located only at vertices), which implies that the number of such computations is $O(n)$ for a fixed p. Thus, up to $\Theta_{\mathrm{OPT}}(k,1,n)$, $O(kn)$ computations are done in total. Using the formulae (8) and (9), the authors of [33] made each computation possible in $O(\log n)$ time, and the same authors have improved it to $O(1)$ time in [35]. They thus developed an $O(kn \log n)$ time algorithm in [33] and an $O(kn)$ time algorithm in [35]. Although all of the above are results for the fractional flow model, it was pointed out in [35] that all properties are still maintained even in the integral flow model; therefore we can directly apply the same algorithms to the integral flow model without increasing time complexity.

Theorem 7 [35] *The minimax k-facility location problem in a dynamic flow path network with uniform edge capacity can be solved in $O(kn)$ time.*

Even for the case of general edge capacities, we now observe that the exactly same approach as in the uniform edge capacity case works although the author of [28] developed an $O(kn^2 \log n)$ time algorithm as shown in Theorem 6. Therefore, we have an $O(kn^2)$ time algorithm since $\Theta_{i,j}(x)$ for fixed i, j and x can be computed in $O(n)$ time using the formulae (10) and (11).

Theorem 8 *The minimax k-facility location problem in a dynamic flow path network with general edge capacities can be solved in $O(kn^2)$ time.*

Sketch of Algorithms Based on Feasibility Tests First, we mention that the authors of [8] developed a data structure CUE tree to efficiently compute $\Theta^i_L(x)$ and $\Theta^j_R(x)$ for any integers i, j satisfying $1 \le i \le j \le n$ 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 using the CUE tree, $\Theta^i_L(x)$ and $\Theta^j_R(x)$ can be computed in $O(\log^2 n)$ time. See [8] for more detail.

Lemma 3 [8] *Given a dynamic flow path network with general edge capacities P, its CUE tree can be constructed in $O(n \log n)$ time.*

Lemma 4 [8] *Given a dynamic flow path network with general edge capacities P, suppose that its CUE tree is available. Then, for any integers i, j satisfying $1 \le i \le j \le n$ and any $x \in P_{i,j}$, $\Theta^i_L(x)$ and $\Theta^j_R(x)$ can be computed in $O(\log^2 n)$ time.*

In [8], to solve the minimax k-facility location problem in a dynamic flow path network, an algorithm repeatedly solves the decision version of the problem for several subpaths. For a positive real t and an integer p, i, j satisfying $1 \le p \le k$ and $1 \le i \le j \le n$, a subpath $P_{i,j}$ is (t, p)-feasible if and only if $\Theta_{\mathrm{OPT}}(p, i, j) \le t$ holds.

Lemma 5 [8] *Given a dynamic flow path network with general edge capacities P, suppose that its CUE tree is available. For integers p, i, j satisfying $1 \le p \le k$ and $1 \le i \le j \le n$, we can test (t, p)-feasibility of $P_{i,j}$ in $O(\min\{n \log^2 n, k \log^3 n\})$ time.*

Proof To determine (t, p)-feasibility of $P_{i,j}$, we put sinks one by one from left to right as far to the right as possible. First, we compute the maximum integer h such that $\Theta^i_L(v_h) \le t$ and $\Theta^i_L(v_{h+1}) > t$ holds. Next, we solve

$$\Theta^i_L(v_{h+1}) - \alpha\tau(v_{h+1} - v_h) = t \tag{37}$$

for α. If $\alpha < 1$, we put the leftmost sink s at $v_{h+1} - \alpha(v_{h+1} - v_h) = \alpha v_h + (1 - \alpha)v_{h+1}$; otherwise we put s at v_h. We then compute the maximum integer l_1 such that $\Theta^{l_1}_R(s) \le t$ and $\Theta^{l_1+1}_R(s) > t$ holds. We thus determine the maximal subpath P_{i,l_1} such that $\Theta_{\mathrm{OPT}}(1, i, l_1) \le t$. In the same manner, we repeatedly isolate the maximal subpaths $P_{l_1,l_2}, P_{l_2,l_3}, \ldots$. If the p-th subpath is determined so that $l_p < j$, $P_{i,j}$ is not (t, p)-feasible; otherwise $P_{i,j}$ is (t, p)-feasible.

Let us turn to the time complexity. When isolating P_{i,l_1}, we did (a) compute h, (b) solve the equation for α, and (c) compute l_1. Obviously (b) takes $O(1)$ time. For (a), if we apply the binary search, it takes $O(\log^3 n)$ time since we compute $O(\log n)$ of $\Theta^i_L(v_a)$ over $i \le a \le j$ and each $\Theta^i_L(v_a)$ can be computed in $O(\log^2 n)$ time using the CUE tree by Lemma 4. Similarly (c) takes $O(\log^3 n)$ time by binary search. In this way we can isolate at most p subpaths in $O(p \log^3 n) = O(k \log^3 n)$ time. On the other hand, if we do not apply the binary search for (a) and (c), i.e., we compute $\Theta^i_L(v_a)$ for $a = i, i+1, \ldots, h, h+1$ and $\Theta^b_R(s)$ for $b = h+1, h+2, \ldots, l_1, l_1+1$, it takes $O((l_1 - i)\log^2 n)$ time to determine P_{i,l_1}. In this way we can isolate at most p subpaths in $O((j - i)\log^2 n) = O(n \log^2 n)$ time. □

Lemma 6 [8] *Given a dynamic flow path network with general edge capacities P, suppose that its CUE tree is available. For any integers i, j satisfying $1 \le i \le j \le n$, $\Theta_{\mathrm{OPT}}(1, i, j)$ can be computed in $O(\log^3 n)$ time.*

Proof Recall that $\Theta_{i,j}(x)$ is unimodal in x (also see (5)). Therefore, for the maximum integer h satisfying $i \le h \le j$ and $\Theta^i_L(v_h) \le \Theta^j_R(v_h)$, we can see that there exists x^* minimizing $\Theta_{i,j}(x)$ on edge e_h including v_h and v_{h+1}. We can apply the binary search to compute such h, which can be done in $O(\log^3 n)$ time using the CUE tree (see Lemma 4). Once h is determined, x^* can be computed as follows: we solve

$$\Theta^i_L(v_{h+1}) - \alpha\tau(v_{h+1} - v_h) = \Theta^j_R(v_h) - (1-\alpha)\tau(v_{h+1} - v_h) \tag{38}$$

for α in $O(1)$ time. If $\alpha \le 0$, let $x^* = v_{h+1}$ and compute $\Theta_{\mathrm{OPT}}(1, i, j) = \Theta_{i,j}(v_{h+1})$; if $\alpha \ge 1$, let $x^* = v_h$ and compute $\Theta_{\mathrm{OPT}}(1, i, j) = \Theta_{i,j}(v_h)$; otherwise, let $x^* = \alpha v_h + (1-\alpha)v_{h+1}$ and compute $\Theta_{\mathrm{OPT}}(1, i, j) = \Theta^i_L(v_{h+1}) - \alpha\tau(v_{h+1} - v_h) = \Theta^j_R(v_h) - (1-\alpha)\tau(v_{h+1} - v_h)$. Using the CUE tree, we can compute such values in $O(\log^2 n)$ time. Thus $\Theta_{\mathrm{OPT}}(1, i, j)$ can be computed in $O(\log^3 n) + O(1) + O(\log^2 n) = O(\log^3 n)$ time. □

To compute the optimal k-sink location on P, the authors of [8] showed two approaches: the parametric search approach and the sorted matrix approach.

In the parametric search approach, we first compute the maximum integer i_1 such that $P_{i_1+1,n}$ is not $(\Theta_{\mathrm{OPT}}(1, 1, i_1), k-1)$-feasible and store $t_1 = \Theta_{\mathrm{OPT}}(1, 1, i_1 + 1)$ as a feasible value. Note that $t^* = \Theta_{\mathrm{OPT}}(k, 1, n)$ satisfies $\Theta_{\mathrm{OPT}}(1, 1, i_1) < t^* \le t_1$. To compute i_1, we apply the binary search by executing $O(\log n)$ tests for $(\Theta_{\mathrm{OPT}}(1, 1, a), k-1)$-feasibility of $P_{a+1,n}$ over $1 \le a \le n$. For an integer a, we can compute $\Theta_{\mathrm{OPT}}(1, 1, a)$ in $O(\log^3 n)$ time by Lemma 6. Also by Lemma 5, we can test if $P_{a+1,n}$ is $(\Theta_{\mathrm{OPT}}(1, 1, a), k-1)$-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 $P_{i_2+1,n}$ is not $(\Theta_{\mathrm{OPT}}(1, i_1 + 1, i_2), k-2)$-feasible and store $t_2 = \Theta_{\mathrm{OPT}}(1, i_1 + 1, i_2 + 1)$ as a feasible value, which can be done in $O(k \log^4 n)$ time in the same manner as in computing (i_1, t_1). Sequentially, we determine $(i_3, t_3), \ldots, (i_{k-1}, t_{k-1})$ in $(k-3) \times O(k \log^4 n)$ time and eventually compute $t_k = \Theta_{\mathrm{OPT}}(1, i_{k-1} + 1, n)$ in $O(\log^3 n)$ time. Notice that $t^* = \min\{t_i \mid i = 1, 2, \ldots, k\}$ holds, which can be computed in $O(k)$ time. We then execute a (t^*, k)-feasibility test for P in $O(k \log^3 n)$ time, so that the optimal k-facility location is obtained. We thus see 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 3, the total time complexity is $O(n \log n + k^2 \log^4 n)$.

In the sorted matrix approach, the authors of [8] defined an $n \times n$ matrix A whose entry (i, j) entry is given by

$$A[i, j] = \begin{cases} \Theta_{\mathrm{OPT}}(1, n - i + 1, j) & \text{if } n - i + 1 \le j \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

It is clear that matrix A includes $\Theta_{\mathrm{OPT}}(1, l, r)$ for every pair of integers (l, r) such that $1 \le l \le r \le n$. There exists a pair (l, r) such that $\Theta_{\mathrm{OPT}}(1, l, r) = \Theta_{\mathrm{OPT}}(k, 1, n)$. Then the problem can be formulated as "Find the smallest $A[i, j]$ such that the given problem instance is $(A[i, j], k)$-feasible." Note that we do not actually compute all the elements of A, but element $A[i, j]$ is computed on demand as needed. We also notice that matrix A is a sorted matrix [25], i.e., each row and column of A is sorted in the nondecreasing order. The authors of [8] mentioned that paper [25] had implicitly shown the following lemma.

Lemma 7 [8, 25] *Suppose that $A[i, j]$ can be computed in $f(n)$ time, and feasibility can be tested in $g(n)$ time. Then the minimax k-facility location problem in a dynamic flow path network can be solved in $O(nf(n) + g(n)\log n)$ time.*

Once the CUE tree is constructed, we have $f(n) = O(\log^3 n)$ by Lemma 6 and $g(n) = O(n\log^2 n)$ by Lemma 5; therefore the problem can be solved in $O(n\log^3 n)$ time. Since it takes $O(n\log n)$ time to construct the CUE tree by Lemma 3, the total time complexity is $O(n\log n) + O(n\log^3 n) = O(n\log^3 n)$.

Theorem 9 [8] *The minimax k-facility location problem in a dynamic flow path network with general edge capacities can be solved in $O(\min\{n\log n + k^2\log^4 n, n\log^3 n\})$ time.*

It was also shown in [8] that the above two approaches similarly work even for the case of uniform edge capacity. See [8] for more detail.

Theorem 10 [8] *The minimax k-facility location problem in a dynamic flow path network with uniform edge capacity can be solved in $O(\min\{n + k^2\log^2 n, n\log n\})$ time.*

3.2.3 *k*-Facility Location in Trees

The minimax 1-facility location problems in a dynamic flow tree network have been studied in [28, 32, 34, 41] so far. Recently, Chen and Golin [14, 15] have studied the case of k-facility location. In this section, we mainly introduce the algorithms for the 1-facility location in [28, 32, 34, 41] and mention the results for the k-facility location in [14, 15] because of space limitations. Given a dynamic flow tree network $\mathcal{N} = (T, w, l, c, \tau)$ where $T = (V, E)$ is a tree such that $|V| = n$, the problem requires finding a location of a facility $x \in T$ that minimizes $\Theta(x)$. Here, the notation T is abused to denote the set of all points on T. In the following, we use the notation x to denote $\boldsymbol{x}$ since we consider the 1-facility location.

Sketch of an Algorithm for the 1-Facility Location We first introduce an algorithm developed by Mamada et al. [41] for the minimax 1-facility location problem in a dynamic flow tree network with general edge capacities under the assumption that a facility is located only at a vertex. Letting x^* denote a point in T that minimizes $\Theta(x)$, the algorithm basically maintains a subtree $T' \subseteq T$ that consists of x^*

while reducing T' by cutting a leaf as well as an incident edge at each step. For a leaf v of T', let $p(v)$ denote a unique vertex adjacent to v in T'. The algorithm also maintains an arriving table A_v and a sending table S_v for every leaf v of T': A_v is a function of time θ such that $A_v(\theta)$ represents the sum of the flow rates arriving at v at time θ from adjacent vertices except $p(v)$, and S_v is also a function of time θ such that $S_v(\theta)$ represents the flow rate leaving v at time θ to $p(v)$. For a leaf v of T', let $time(v)$ denote the time at which the last flow sent by S_v arrives at $p(v)$. Then, the algorithm computes $time(v)$ for every leaf v of T' and remove from T' a leaf v^* (and an edge incident to v^*) that minimizes $time(v)$ over all leaves v of T'. If $p(v^*)$ becomes a new leaf of T', the algorithm computes $A_{p(v^*)}$. We here observe that the maximum cost of $p(v^*)$ is less than or equal to that of v^*, i.e., v^* is not an optimal facility location or both of v^* and $p(v^*)$ are optimal. Thus, when T' becomes a single vertex, the algorithm outputs the vertex as x^*. The authors of [41] mentioned that the above algorithm requires $O(n^2)$ time if Arriving and Sending Tables are explicitly constructed, and by representing these tables efficiently, they developed an $O(n \log^2 n)$ time algorithm.

Theorem 11 [41] *The minimax 1-facility location problem in a dynamic flow tree network with general edge capacities can be solved in $O(n \log^2 n)$ time.*

We next overview an $O(n \log n)$ time algorithm for the minimax 1-facility location problem in a dynamic flow tree network with uniform edge capacity by [28, 32, 34]. To develop an algorithm for the case of the uniform edge capacity, the authors of [28, 32, 34] used the formula of $\Theta(x)$ observed by Kamiyama et al. [38], which is as follows. For two points $x, y \in T$, let $d(x, y)$ denote the distance between x and y in T. For a vertex $v \in V$, let $\delta(v)$ denote a set of vertices adjacent to v, and for a point $x \in T$ which is not at a vertex but on an edge $uv \in E$, let $\delta(x)$ denote a set of two vertices u and v. Given a point $x \in T$, if x is not at a vertex but on an edge $uv \in E$, x is regarded as a new vertex of T and uv is split into two new edges ux and xv. Then, let $T(x)$ be a rooted tree made from T such that each edge has a natural orientation towards the root x, and for any vertex $v \in V$, let $T(x, v)$ be a subtree of $T(x)$ rooted at v. Suppose that a facility is located at a point $x \in T$. For a vertex $u \in \delta(x)$, let $\Theta(x, u)$ denote the maximum cost of x for all supplies on $T(x, u)$, i.e., the minimum time required to send all supplies on $T(x, u)$ to x. Then $\Theta(x)$ can be represented as follows:

$$\Theta(x) = \max\{\Theta(x, u) \mid u \in \delta(x)\}. \tag{40}$$

Here, we only need to consider $\Theta(x, u^*)$ for $u^* = \operatorname{argmax}\{\Theta(x, u) \mid u \in \delta(x)\}$. Suppose that there are n' vertices in $T(x, u^*)$ named $v_1 (= u^*), v_2, \ldots, v_{n'}$ such that $d(x, v_j) \le d(x, v_{j+1})$ for $1 \le j \le n' - 1$. For the integral flow model, Kamiyama et al. [38] observed that the value of $\Theta(x, u^*)$ does not change if x and all v_j for $1 \le j \le n'$ are relocated on a line with the same capacity so that $d(x, v_j)$ for $1 \le j \le n'$ remain the same (see Fig. 3), and $\Theta(x, u^*)$ can be represented as follows:

$$\Theta(x, u^*) = \max\left\{ \tau d(x, v_i) + \left\lceil \frac{\sum_{i \le j \le n'} w(v_j)}{c'} \right\rceil - 1 \;\middle|\; 1 \le i \le n' \right\}, \tag{41}$$

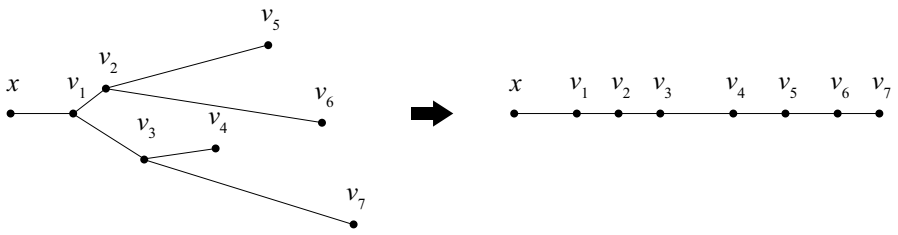

Fig. 3 Vertices of the tree can be relocated on a line with the same capacity

where c' is the uniform edge capacity. The authors of [28, 32, 34] have formally proved the formula (41), and the author of [28] also developed the formula for the fractional flow model as follows:

$$\Theta(x, u^*) = \max\left\{ \tau d(x, v_i) + \frac{\sum_{i \le j \le n'} w(v_j)}{c'} \;\middle|\; 1 \le i \le n' \right\}. \tag{42}$$

Also, to solve the problem effectively, the authors of [28, 32, 34] proved key properties shown in Lemmas 8 and 9. In the statements and the proofs of those two lemmas, we use the following notation. For two vertices $v, v' \in V$, let $V(v, v')$ denote the set of all vertices in $T(v, v')$ and let $T(V')$ denote a subgraph induced by a vertex set $V' \subseteq V$. Let P be a simple path in T from a leaf to another leaf, which is represented as the sequence of vertices $v_1, v_2, \ldots, v_k$ where v_1 and v_k are leaves. In the following, for a point $x \in P$, the notation x is abused to denote $d(v_1, x)$. For a point $x \in P$, we call the direction to v_1 (resp. v_k) from x the left direction (resp. right direction). If a facility location x is given at a vertex v_i for some i satisfying $2 \le i \le k$ (resp. $1 \le i \le k-1$), let $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) denote the minimum time required to send all supplies on $T(x, v_{i-1})$ (resp. $T(x, v_{i+1})$) to x. If x is given on an edge $v_i v_{i+1}$ for some i satisfying $1 \le i \le k-1$, let $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) denote the minimum

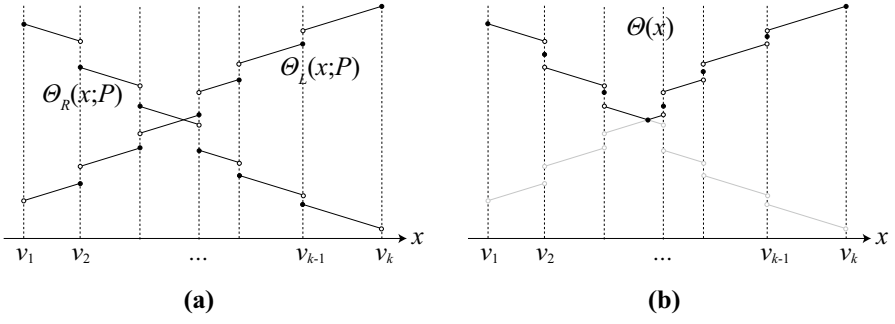

Fig. 4 Functions along P: **a** $\Theta_L(x;P)$, $\Theta_R(x;P)$ and **b** $\Theta(x)$

time required to send all supplies on $T(x, v_i)$ (resp. $T(x, v_{i+1})$) to x. Also, for a vertex v_i satisfying $1 \leq i \leq k-1$, let

$$\Theta_L^{+0}(v_i;P) = \lim_{\epsilon \to +0} \left\{ \Theta_L(v_i + \epsilon;P) \right\}, \tag{43}$$

$$\Theta_R^{-0}(v_i;P) = \lim_{\epsilon \to +0} \left\{ \Theta_R(v_i - \epsilon;P) \right\}. \tag{44}$$

Lemma 8 *Along a path from a leaf to another leaf in T, $\Theta(x)$ is unimodal in x.*

Proof To prove the lemma, we first show that along a path P, $\Theta_L(x;P)$ is increasing in x and $\Theta_R(x;P)$ is decreasing in x (see Fig. 4a). By (42), (43) and (44), we can see the following three properties of $\Theta_L(x;P)$ and $\Theta_R(x;P)$:

(i) for an open interval (v_{i-1}, v_i) satisfying $2 \leq i \leq k$, $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is linear in x with slope τ (resp. $-\tau$),
(ii) $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is left-continuous (resp. right-continuous) at $x = v_i$ for $2 \leq i \leq k$ (resp. $1 \leq i \leq k-1$),
(iii) $\Theta_L(v_i;P) \leq \Theta_L^{+0}(v_i;P)$ (resp. $\Theta_R^{-0}(v_i;P) \geq \Theta_R(v_i;P)$) holds at v_i for $2 \leq i \leq k-1$.

From these properties, $\Theta_L(x;P)$ (resp. $\Theta_R(x;P)$) is piecewise linear increasing (resp. decreasing) in x. Therefore, there uniquely exists $x \in P$ that minimizes $\max\{\Theta_L(x;P), \Theta_R(x;P)\}$, called $x^*(P)$ in the following.

Also, it can be seen that $\Theta_L(v_i;P) \leq \Theta(v_i) \leq \Theta_L^{+0}(v_i;P)$ for a vertex $v_i \in P$ such that $v_i \geq x^*(P)$, and $\Theta_R^{-0}(v_i;P) \geq \Theta(v_i) \geq \Theta_R(v_i;P)$ for a vertex $v_i \in P$ such that $v_i \leq x^*(P)$. Thus, $\Theta(x)$ may possibly be discontinuous at v_i for $2 \leq i \leq k-1$ but it is always unimodal in x along P. □

Lemma 9 *For a vertex $v \in V$, if $u^* = \operatorname{argmax}\{\Theta(v,u) \mid u \in \delta(v)\}$ holds, there exists $x^* \in T(V(v,u^*) \cup \{v\})$.*

Proof Let us consider a path P from a leaf to another leaf through adjacent vertices v and u^* where $u^* = \operatorname{argmax}\{\Theta(v,u) \mid u \in \delta(v)\}$. Let us define the left direction in P as the direction from v to u^* and the right direction as the other one. Suppose that there are $k+1$ vertices $v_1, v_2, \ldots, v_k$ in P, and $v = v_i$ and $u^* = v_{i-1}$ for some i satisfying $2 \leq i \leq k-1$. We consider a point $p \in P$ such that $p = v_i + \epsilon$ with sufficiently small $\epsilon > 0$.

If we can show $\Theta(v_i) < \Theta(p)$, x^* never exists in the right direction from v_i along P by Lemma 8. Then, this lemma can be proved by repeatedly applying the same discussion to all the other paths through v and u^*. By the assumption of $\Theta(v_i) = \Theta_L(v_i;P)$,

$\Theta_L(v_i;P) \geq \Theta_R(v_i;P)$ holds, and by (42), $\Theta_L(v_i;P) + \epsilon\tau \leq \Theta_L(p;P)$ and $\Theta_R(v_i;P) = \Theta_R(p;P) + \epsilon\tau$, that is, $\Theta_L(v_i;P) < \Theta_L(p;P)$ and $\Theta_R(v_i;P) > \Theta_R(p;P)$ hold. Thus, we have $\Theta_R(p;P) < \Theta_L(p;P)$, which implies that

$$\Theta(p) = \Theta_L(p;P). \tag{45}$$

From (45) and the above mentioned two facts $\Theta(v_i) = \Theta_L(v_i;P)$ and $\Theta_L(v_i;P) < \Theta_L(p;P)$, we derive $\Theta(v_i) < \Theta(p)$.

□

We now see the algorithm by [28, 32, 34] for the minimax 1-facility location problem in a dynamic flow tree network with uniform edge capacity. The algorithm uses the concept of centroid of a tree [40]: for an undirected tree $T = (V, E)$, a centroid of T is a vertex that minimizes $\max\{|V(v,u)| \mid u \in \delta(v)\}$ for all $v \in V$.

Kang et al. [40] showed that a centroid m of T can be computed in $O(|V|)$ time and

$$\max\{|V(m,u)| \mid u \in \delta(m)\} \leq \frac{|V|}{2}. \tag{46}$$

Let us explain the first iteration of the algorithm by [28, 32, 34]. For ease of explanation, we here assume that a facility is located on a vertex although the authors of [28, 32, 34] allowed that a facility can be located at any point on a tree. Letting $U_1 = V$, the algorithm first finds a centroid m_1 of $T(U_1)$ and computes $d(m_1, v)$ for every $v \in U_1$. Then, to compute $\Theta(m_1, u)$ for each $u \in \delta(m_1)$, the algorithm basically creates the list $L(u)$ of all vertices $v \in U_1 \cap V(m_1, u)$ which are arranged in the nondecreasing order of $d(m_1, v)$. From (42), we can derive that $\Theta(m_1, u)$ can be computed using $L(u)$.

In this manner, the algorithm computes $u_1 = \text{argmax}\{\Theta(m_1, u) \mid u \in \delta(m_1)\}$. After that, it sets $V_1 = U_1 \setminus (V(m_1, u_1) \cup \{m_1\})$ and merges lists $L(u)$ for $u \in \delta(m_1) \setminus \{u_1\}$ into a new list L_1. At the end of the first iteration, the algorithm sets $U_2 = U_1 \cap (V(m_1, u_1) \cup \{m_1\})$. Note that by Lemma 9, there exists x^* in $T(U_2)$ and by (46), $|U_2| \leq |U_1|/2 + 1$ holds.

The algorithm iteratively performs the same procedure (see Fig. 5). Namely, at the i-th iteration, it finds a centroid m_i of $T(U_i)$, computes $u_i = \text{argmax}\{\Theta(m_i, u) \mid u \in \delta(m_i)\}$, sets $V_i = U_i \setminus (V(m_i, u_i) \cup \{m_i\})$, creates a list L_i of vertices $v \in V_i$ arranged in the nondecreasing order of $d(m_i, v)$ and also sets $U_{i+1} = U_i \cap (V(m_i, u_i) \cup \{m_i\})$. Since, at each iteration, the algorithm reduces the subgraph where x^* exists so that the size becomes half or less roughly, it halts after $l = O(\log |V|)$ iterations. At this point, it finds two vertices m_l and $u_l \in U_l$ connected by an edge, and outputs the better one as x^*.

Let us turn to the time complexity of the algorithm. The authors of [28, 32, 34] first showed that the running time is $O(n \log^2 n)$ where $n = |V|$. Let us examine the running time for each iteration required by the algorithm. At the i-th iteration for $i \geq 2$, a centroid m_i of $T(U_i)$ can be found in $O(|U_i|)$ time (in [40]), and $d(m_i, v)$ can be computed for all $v \in V$ by depth-first search in $O(n)$ time. In the following, we consider two lists of vertices in $V(m_i, u)$ for $u \in \delta(m_i)$ which are arranged in the nondecreasing order of the distance from m_i, that is, $L(u)$ and $L'(u)$. The only one difference between $L(u)$ and $L'(u)$ is that $L(u)$ just consists of vertices in $U_i \cap V(m_i, u)$ although $L'(u)$ consists of all vertices in $V(m_i, u)$. If the algorithm creates a list $L'(u)$, $\Theta(m_i, u)$ can be computed as mentioned above. Each list

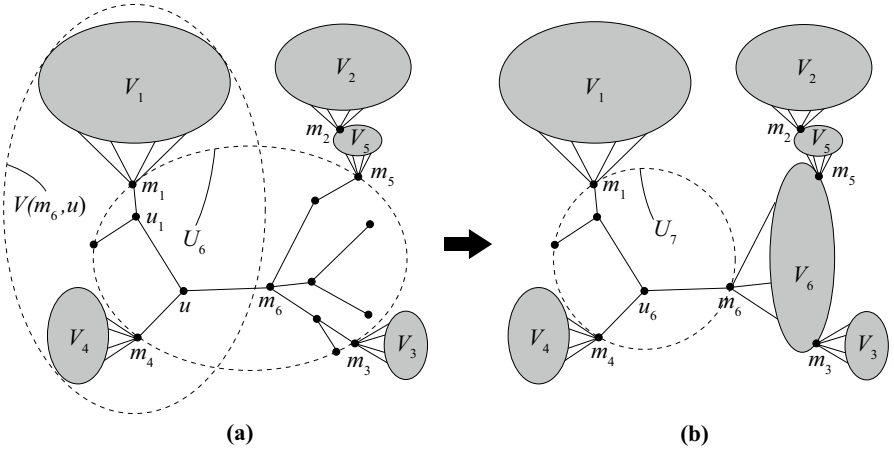

Fig. 5 Illustration of the i-th iteration: **a** $i = 6$ and **b** $i = 7$

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

In the above analysis, the authors of [28, 32, 34] showed that the running time required to create lists $L'(u)$ for $u \in \delta(m_i)$ can be improved from $O(n \log n)$ to $O(n + |U_i| \log |U_i|)$, based on the following lemma.

Lemma 10 *$|U_i| = O(\frac{n}{2^{i-1}})$ and $|V_i| = O(\frac{n}{2^{i-1}})$ hold for $i \geq 1$.*

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

The idea to improve the running time is to use the sorted lists L_j with $j = 1, 2, \ldots, i-1$. Let us look at Fig. 5a, and focus on a vertex $u \in \delta(m_6)$ in the figure. The computation of $L'(u)$ can be done in $O(n \log n)$ time if we know $d(m_6, v)$ for all $v \in V(m_6, u)$. But, since $V(m_6, u) = V_1 \cup V_4 \cup (U_6 \cap V(m_6, u))$ holds and we have already computed L_1 and L_4, $L'(u)$ can be obtained faster if we create only a list $L(u)$ by computing $d(m_6, v)$ for all $v \in U_6 \cap V(m_6, u)$. Note that by (46), $|U_6 \cap V(m_6, u)|$ is at most $|U_6|/2$, which is about $|V_1|/64$ or $|V_4|/8$ by Lemma 10, so its size is much smaller than $|V(m_6, u)|$. The idea is formalized as follows. For each $u \in \delta(m_i)$, the algorithm first creates a list $L(u)$ of vertices in $U_i \cap V(m_i, u)$, which takes $O(n' \log n')$ time where $n' = |U_i \cap V(m_i, u)|$. Thus, lists $L(u)$ for all $u \in \delta(m_i)$ can be created in $O(|U_i| \log |U_i|)$ time. For each $u \in \delta(m_i)$, the algorithm merges $L(u)$ and all lists L_j with $V_j \subseteq V(m_i, u)$ into $L'(u)$ (at this point, all

of the original lists are maintained since these will be used later). For this merging operation, if we apply a simple merge sort, it takes $O(|V(m_i, u)| \log |V(m_i, u)|)$ time, which does not improve the running time. Here, we notice that $|L_j| = |V_j|$ for $1 \le j \le i-1$. Instead, the algorithm basically takes the following two steps to create each list $L'(u)$ for $u \in \delta(m_i)$.

[Step 1]: For L_j such that $V_j \subseteq V(m_i, u)$, choose $L_p = \text{argmin}\{|L_j| \mid V_j \subseteq V(m_i, u)\}$ and merge each L_j in the increasing order of size (i.e., the decreasing order of j) with L_p one by one.

[Step 2]: Merge the list obtained at Step 1 and $L(u)$ into $L'(u)$.

For all $u \in \delta(m_i)$, Step 1 takes in $O(\sum_{j=1}^{i-1} jn/2^{j-1}) = O(n)$ time, and thus, Step 2 takes $O(n + |U_i|) = O(n)$ time. Recall that $L(u)$ for all $u \in \delta(m_i)$ can be created in $O(|U_i| \log |U_i|)$ time. Then, by Lemma 10, it takes $O(n + |U_i| \log |U_i|) = O(n + (n/2^{i-1}) \log(n/2^{i-1}))$ time to create lists $L'(u)$ for all $u \in \delta(m_i)$.

Lemma 11 *The i-th iteration of the algorithm by* [28, 32, 34] *takes* $O(n + \frac{n}{2^{i-1}} \log \frac{n}{2^{i-1}})$ *time.*

Recall that the algorithm halts after $O(\log n)$ iterations. Thus, by Lemma 11, it takes $O(n \log n + \sum\{(n/2^{i-1}) \log(n/2^{i-1}) \mid 1 \le i \le \log n\}) = O(n \log n)$ time for the entire iterations.

Theorem 12 [28, 32, 34] *The minimax 1-facility location problem in a dynamic flow tree network with uniform edge capacity can be solved in* $O(n \log n)$ *time.*

Results for the k-Facility Location We here introduce the latest results in [15] for the minimax k-facility location problem in a dynamic flow tree network. Note that the algorithms by [15] use the algorithms for the 1-facility location by [28, 32, 34, 41] as subroutines.

Theorem 13 [15] *The minimax k-facility location problem in a dynamic flow tree network with general edge capacities can be solved in* $O(\max\{k, \log n\} kn \log^4 n)$ *time. The problem for the case of uniform edge capacity can be solved in* $O(\max\{k, \log n\} kn \log^3 n)$ *time.*

3.3 Minisum Facility Location Problems

We consider the facility location problems in a dynamic flow network with the integral flow model. Given a location of facilities $\boldsymbol{x}$ and a feasible evacuation to $\boldsymbol{x}$, say $\mathscr{E}$, the cost of $(\boldsymbol{x}, \mathscr{E})$ for an evacuee is defined as the time required to send him/her to a facility of $\boldsymbol{x}$ along an evacuation path determined by $\mathscr{E}$. Then $sum(\boldsymbol{x}, \mathscr{E})$ is defined as the sum of the cost of $(\boldsymbol{x}, \mathscr{E})$ for all evacuees. The total cost of $\boldsymbol{x}$ is defined as

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \min\{sum(\boldsymbol{x}, \mathscr{E}) \mid \mathscr{E} \in \mathfrak{E}_x\}, \tag{47}$$

where $\mathfrak{E}_x$ is a set of all feasible evacuations to $\boldsymbol{x}$. In the fractional flow model, we define the unit as the infinitesimally small portion of supply, and the cost is defined on each infinitesimal unit. Then two criteria are defined in the same way as in the integral flow model. Here, we deal with a problem that requires finding $\boldsymbol{x}$ in a dynamic flow network minimizing $\boldsymbol{\Phi}(\boldsymbol{x})$.

3.3.1 *k*-Facility Location in Paths

The minisum k-facility location problems in a dynamic flow path network with uniform edge capacity have been studied in [33, 35]. Very recently, Benkoczi et al. [7] have studied the problem for the case of general edge capacities. In this section, we mainly introduce the algorithms for the case of uniform edge capacity in [33, 35] and mention the results for the case of general edge capacities in [7] because of space limitations. Given a dynamic flow path network $\mathcal{N} = (P = (V, E), w, l, c, \tau)$ defined in Sect. 3.2.2, the problem requires finding a location of k facilities $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in P^k$ that minimizes $\boldsymbol{\Phi}(\boldsymbol{x})$. In [7, 33, 35], it was assumed that all units of supply originally located at a vertex are sent to the same facility (as in the maximum cost case). In the following, we use the notation x to denote $\boldsymbol{x}$ when we consider the 1-facility location.

Basic Properties for the Case of Uniform Edge Capacity We first see a useful formula developed by [33, 35] for the total cost in the case of uniform edge capacity to solve the 1-facility location problem. Suppose that a facility is located at a point $x \in P_{i,j}$ for fixed integers i and j satisfying $1 \le i \le j \le n$. Let $\boldsymbol{\Phi}_{i,j}(x)$ denote the total cost of x for $P_{i,j}$. Also let $\boldsymbol{\Phi}^i_L(x)$ (resp. $\boldsymbol{\Phi}^j_R(x)$) denote the sum of the cost of x for all supplies on the part of $P_{i,j}$ between v_i and x (resp. x and v_j) where $\boldsymbol{\Phi}^i_L(v_i) = 0$ and $\boldsymbol{\Phi}^j_R(v_j) = 0$. Then, $\boldsymbol{\Phi}_{i,j}(x)$ is the sum of $\boldsymbol{\Phi}^i_L(x)$ and $\boldsymbol{\Phi}^j_R(x)$, i.e.,

$$\boldsymbol{\Phi}_{i,j}(x) = \boldsymbol{\Phi}^i_L(x) + \boldsymbol{\Phi}^j_R(x). \tag{48}$$

For the fractional flow model, the authors of [33, 35] showed the formulae for $\boldsymbol{\Phi}^i_L(x)$ and $\boldsymbol{\Phi}^j_R(x)$. In the following, we only explain the case of $\boldsymbol{\Phi}^i_L(x)$ (the case of $\boldsymbol{\Phi}^j_R(x)$ is symmetric). Assume that x is located on an edge e_{s_x} (not including endpoints) satisfying $i \le s_x \le j-1$. Consider the case that the first unit of v_{l-1} is forced to stop at v_l for some l satisfying $i+1 \le l \le s_x$, i.e.,

$$\tau(v_l - v_{l-1}) \le \frac{w(v_l)}{c'}, \tag{49}$$

where c' is the uniform edge capacity. Then, even if all units of v_{l-1} are moved to v_l and v_{l-1} is removed from $P_{i,j}$, the cost of x for any unit does not change, which implies that $\boldsymbol{\Phi}^i_L(x)$ does not change. By repeatedly applying the same operation as long as there exist such consecutive two vertices, several vertices whose indices are $\rho_1, \ldots, \rho_e$ are eventually left such that all units on vertices $v_{\rho_{h-1}+1}, \ldots, v_{\rho_h-1}$ have been moved to v_{ρ_h} for $1 \le h \le e$ (where $\rho_0 = i-1$ and $\rho_e = s_x$) without changing $\boldsymbol{\Phi}^i_L(x)$. Notice that for every integer h satisfying $1 \le h \le e$, the first unit of v_{ρ_h} will never be

forced to stop on its way to x. Formally, the authors of [33, 35] defined the vertex indices $\rho_1, \ldots, \rho_e$ as

$$\rho_i = \text{argmax}\left\{ \tau(x - v_h) + \frac{\sum_{l=\rho_{i-1}+1}^{h} w(v_l)}{c'} \;\middle|\; \rho_{i-1} + 1 \le h \le s_x \right\}. \tag{50}$$

For every integer h satisfying $1 \le h \le e$, they also defined the value of ρ_h as $\sigma_h = \sum\{w(v_l) \mid \rho_{h-1} + 1 \le l \le \rho_h\}$. Then, it was shown in [33, 35] that $\boldsymbol{\Phi}^i_L(x)$ is represented as follows:

$$\boldsymbol{\Phi}^i_L(x) = \sum_{1 \le h \le e} \left(\sigma_h \tau(x - \rho_h) + \frac{{\sigma_h}^2}{2c'} \right). \tag{51}$$

The authors of [33, 35] pointed out that function $\boldsymbol{\Phi}_{i,j}(x)$ may not be unimodal in x although $\boldsymbol{\Phi}^i_L(x)$ and $\boldsymbol{\Phi}^j_R(x)$ are increasing in x and decreasing in x, respectively.

Also, to efficiently solve the problem, the authors of [33, 35] proved an important property.

Lemma 12 *For the minisum 1-facility location problem in a dynamic flow path network with uniform edge capacity, there exists an optimal facility location at a vertex.*

Proof By (51), for an open interval (v_h, v_{h+1}) with $i \le h \le j-1$, $\boldsymbol{\Phi}_{i,j}(x)$ is linear in x with slope $\tau(\sum_{i \le l \le h} w_l - \sum_{h+1 \le l \le j} w_l)$. Now let us consider an open interval (v_h, v_{h+1}) with $i \le h \le j-1$ such that $\sum_{i \le l \le h} w_l - \sum_{h+1 \le l \le j} w_l \ge 0$ holds. Then, we can see that for any two points $p, q \in (v_h, v_{h+1})$ satisfying $p < q$, $\boldsymbol{\Phi}_{i,j}(p) \le \boldsymbol{\Phi}_{i,j}(q)$ holds. We will show that for sufficiently small $\epsilon > 0$, $\boldsymbol{\Phi}_{i,j}(v_h) \le \boldsymbol{\Phi}_{i,j}(v_h + \epsilon)$ holds. We confirm

$$\boldsymbol{\Phi}^j_R(v_h) = \boldsymbol{\Phi}^j_R(v_h + \epsilon) + \left(\sum_{h+1 \le l \le j} w_l \right) \cdot \tau\epsilon, \quad \text{and} \tag{52}$$

$$\boldsymbol{\Phi}^i_L(v_h + \epsilon) \ge \boldsymbol{\Phi}^i_L(v_h) + \left(\sum_{i \le l \le h} w_l \right) \cdot \tau\epsilon. \tag{53}$$

From (52), (53) and the assumption of $\sum_{i \le l \le h} w_l - \sum_{h+1 \le l \le j} w_l \ge 0$, we can derive $\boldsymbol{\Phi}_{i,j}(v_h) \le \boldsymbol{\Phi}_{i,j}(v_h + \epsilon)$. Thus, for any point $p \in (v_h, v_{h+1})$, $\boldsymbol{\Phi}_{i,j}(v_h) \le \boldsymbol{\Phi}_{i,j}(p)$ holds, which implies that $x^*(i,j)$ is located at some vertex where $x^*(i,j) = \text{argmin}\{\boldsymbol{\Phi}_{i,j}(x) \mid x \in P_{i,j}\}$. □

Sketch of Algorithms for the Case of Uniform Edge Capacity We now see the algorithm by [33, 35] for the minisum 1-facility location problem in a dynamic flow path network with uniform edge capacity. For ease of explanation, we consider

$P = P_{1,n}$ as an input, and use the notation $\boldsymbol{\Phi}(x)$, $\boldsymbol{\Phi}_L(x)$ and $\boldsymbol{\Phi}_R(x)$ to denote $\boldsymbol{\Phi}_{1,n}(x)$, $\boldsymbol{\Phi}^1_L(x)$ and $\boldsymbol{\Phi}^n_R(x)$.

Basically, the algorithm first computes $\boldsymbol{\Phi}_L(v_i)$ for $2 \le i \le n$ in ascending order of i, and next $\boldsymbol{\Phi}_R(v_i)$ for $1 \le i \le n-1$ in descending order of i. After computing all these values, $\boldsymbol{\Phi}(v_i)$ can be computed and evaluated for $1 \le i \le n$ in $O(n)$ time. By Lemma 12, the optimal facility location x^* is at a vertex that minimizes $\boldsymbol{\Phi}(x)$ for $x \in P$, so we just take the minimum of the n values above.

Below, we show how to compute $\boldsymbol{\Phi}_L(v_i)$ (computation of $\boldsymbol{\Phi}_R(v_i)$ can be treated in a similar manner). First, the algorithm sets $\rho_1 = 1$, $\sigma_1 = w_1$. By (51), $\boldsymbol{\Phi}_L(v_2)$ is computed in $O(1)$ time as follows:

$$\boldsymbol{\Phi}_L(v_2) = \sigma_1 \tau(v_2 - v_{\rho_1}) + \frac{{\sigma_1}^2}{2c'}. \tag{54}$$

Now, suppose that for some integer j satisfying $2 \le j \le n-1$, $\boldsymbol{\Phi}_L(v_j)$ has been already obtained as follows:

$$\boldsymbol{\Phi}_L(v_j) = \sum_{1 \le i \le e(j)} \left(\sigma_i \tau(v_j - v_{\rho_i}) + \frac{{\sigma_i}^2}{2c'} \right), \tag{55}$$

where $e(j)$ is a positive integer satisfying $1 \le e(j) \le j-1$. Here, ρ_i and σ_i for all i satisfying $1 \le i \le e(j)$ have also been obtained. In addition, letting $W_{j-1} = \sum_{1 \le i \le j-1} w_i = \sum_{1 \le i \le e(j)} \sigma_i$, suppose that W_{j-1} has also been computed. We then show how to compute $\boldsymbol{\Phi}_L(v_{j+1})$. First, the algorithm temporarily sets

$$\boldsymbol{\Phi}' = \boldsymbol{\Phi}_L(v_j), \quad \text{and} \quad W' = W_{j-1}. \tag{56}$$

Next, the algorithm computes W_j as $W_j = W_{j-1} + w_j$. Here, $W_j - W'$ corresponds to w_j plus the amount of supplies merged to v_j. Then the algorithm tests if $\tau(v_j - v_{\rho_i}) < (W_j - W')/c'$ for $1 \le i \le e(j)$ in descending order. If and only if so, the supply of v_{ρ_i} will be merged to v_j, thus the algorithm updates $\boldsymbol{\Phi}'$ and W' as follows:

$$\boldsymbol{\Phi}' \leftarrow \boldsymbol{\Phi}' - \left(\sigma_i \tau(v_j - v_{\rho_i}) + \frac{{\sigma_i}^2}{2c'} \right), \quad \text{and} \quad W' \leftarrow W' - \sigma_i, \tag{57}$$

and deletes ρ_i. If the maximum integer m such that $\tau(v_j - v_{\rho_m}) \ge (W_j - W')/c'$ is found or $\tau(v_j - v_{\rho_1}) < (W_j - W')/c'$ is obtained, the algorithm stops testing. In the former case, after the algorithm tests $e(j) - m + 1$ times, $\rho_1, \ldots, \rho_m$ remain. Now the total amount of supplies on $v_{\rho_1}, \ldots, v_{\rho_m}$ is W', and the total evacuation time to v_j for these supplies is $\boldsymbol{\Phi}'$. Since each unit of these supplies does not stop at v_j, the total evacuation time to v_{j+1} for these supplies is $\boldsymbol{\Phi}' + W'\tau(v_{j+1} - v_j)$. On the other hand, all supplies on $v_{\rho_{m+1}}, \ldots, v_{\rho_{e(j)}}$ are merged to v_j, thus the total amount of supply on v_j is $W_j - W'$. Then, by (51), $\boldsymbol{\Phi}_L(v_{j+1})$ can be computed as

$$\begin{aligned} \boldsymbol{\Phi}_L(v_{j+1}) = \boldsymbol{\Phi}' &+ W'\tau(v_{j+1} - v_j) \\ &+ \left((W_j - W')\tau(v_{j+1} - v_j) + \frac{(W_j - W')^2}{2c'} \right). \end{aligned} \tag{58}$$

Also, for the next recursive step, the algorithm eventually sets

$$e(j+1) = m+1, \quad \rho_{m+1} = j, \quad \text{and} \quad \sigma_{m+1} = W_j - W'. \tag{59}$$

Since the algorithm tests $e(j) - m + 1 = e(j) - e(j+1) + 2$ times to compute $\boldsymbol{\Phi}_L(v_{j+1})$, it needs to test $\sum_{1 \le i \le n-1}(e(i) - e(i+1) + 2)$ times to compute $\boldsymbol{\Phi}_L(v_i)$ for $2 \le i \le n$. Here, by $e(1) = 0$, we have

$$\sum_{1 \le i \le n-1} (e(i) - e(i+1) + 2) = -e(n) + 2(n-1) = O(n). \tag{60}$$

Theorem 14 [33, 35] *The minisum 1-facility location problem in a dynamic flow path network with uniform edge capacity can be solved in $O(n)$ time.*

For the minisum k-facility location problem in a dynamic flow path network with uniform edge capacity, Higashikawa et al. [33] showed the following recursion for $p \ge 2$ in a similar way to (12):

$$\boldsymbol{\Phi}_{\mathrm{OPT}}(p,i,j) = \min_{i \le d \le j-1} \{\boldsymbol{\Phi}_{\mathrm{OPT}}(p-1,i,d) + \boldsymbol{\Phi}_{\mathrm{OPT}}(1,d+1,j)\}, \tag{61}$$

where $\boldsymbol{\Phi}_{\mathrm{OPT}}(p,i,j)$ denotes the cost of the minisum p-facility location in a subpath $P_{i,j}$ (which has already been defined in Sect. 3.2.2). Also letting $d_p(j)$ denote an integer t minimizing $\{\boldsymbol{\Phi}_{\mathrm{OPT}}(p-1,1,t) + \boldsymbol{\Phi}_{\mathrm{OPT}}(1,t+1,j)\}$ for $1 \le t \le j-1$, the authors of [33] showed that for integers p, j satisfying $2 \le p \le k$ and $1 \le j \le n-1$, $d_p(j) \le d_p(j+1)$ holds (similar to Lemma 1). Using a dynamic programming approach based on the above property, the authors of [33] proved that the minisum k-facility location problem can be solved by computing the minisum 1-facility location problems in subpaths for $O(kn)$ times. By this and Theorem 14, they developed an $O(kn^2)$ algorithm.

Next, the same authors as [33] have improved the algorithm in its time complexity in [35]. They basically transformed the problem to an equivalent problem, which requires finding the minimum k-link path in a weighted, complete, directed acyclic graph (DAG), as follows. First, for integers i and j satisfying $1 \le i < j \le n+1$, let $w(i,j) = \boldsymbol{\Phi}_{\mathrm{OPT}}(1,i,j-1)$. Let us consider a DAG $G = (N, A)$ such that $N = \{u_1, u_2, \ldots, u_n, u_{n+1}\}$ and for every vertex pair (u_i, u_j) satisfying $1 \le i < j \le n+1$, there exists an edge which is directed from u_i to u_j associated with the weight of $w(i, j)$. Then, the minisum k-facility location problem in a dynamic flow path network is equivalent to a problem requiring finding a path in G from u_1 to u_{n+1} which contains exactly k edges such that the sum of weights is minimized. Schieber [43] showed that this problem can be solved by querying edge weights $\min\{O(n\sqrt{k \log n} + n \log n), n2^{O(\sqrt{\log k \log\log n})}\}$ times if the input DAG satisfies the concave Monge property, that is, $w(i,j) + w(i+1,j+1) \le w(i+1,j) + w(i,j+1)$ holds for any integers i and j satisfying $2 \le i+1 < j \le n$. The authors of [35] proved that the concave Monge property holds if $w(i,j) = \boldsymbol{\Phi}_{\mathrm{OPT}}(1,i,j-1)$. Then, we can see a $\min\{O(n^2\sqrt{k \log n} + n^2 \log n), n^2 2^{O(\sqrt{\log k \log\log n})}\}$ time algorithm for the minisum k-facility location problem in a dynamic flow path network since each weight query takes $O(n)$ time as mentioned in Theorem 14.

Lemma 13 *For any integers i and j satisfying* $2 \le i+1 < j \le n$,

$$\Phi_{\mathrm{OPT}}(1,i,j-1) + \Phi_{\mathrm{OPT}}(1,i+1,j) \le \Phi_{\mathrm{OPT}}(1,i+1,j-1) + \Phi_{\mathrm{OPT}}(1,i,j).$$

Proof We consider two cases: [Case 1] $x^*(i+1,j-1) \le x^*(i,j)$ and [Case 2] $x^*(i+1,j-1) > x^*(i,j)$. We will only prove Case 1 (The proof of Case 2 is symmetric). We first show that

$$\begin{aligned} \Phi_{i,j-1}(x^*(i+1,j-1)) - \Phi_{i+1,j-1}(x^*(i+1,j-1)) \\ \le \Phi_{i,j}(x^*(i,j)) - \Phi_{i+1,j}(x^*(i,j)). \end{aligned} \tag{62}$$

Since both of $\Phi_{i,j-1}(x^*(i+1,j-1))$ and $\Phi_{i+1,j-1}(x^*(i+1,j-1))$ include $\Phi_R^{j-1}(x^*(i+1,j-1))$, the left side of (62) is equal to $\Phi_L^i(x^*(i+1,j-1)) - \Phi_L^{i+1}(x^*(i+1,j-1))$. Similarly, the right side of (62) is equal to $\Phi_L^i(x^*(i,j)) - \Phi_L^{i+1}(x^*(i,j))$. Let $D = \Phi_L^{i+1}(x^*(i,j)) - \Phi_L^{i+1}(x^*(i+1,j-1))$ (clearly $D \ge 0$ by the condition of $x^*(i+1,j-1) \le x^*(i,j)$), that is,

$$\Phi_L^{i+1}(x^*(i,j)) = \Phi_L^{i+1}(x^*(i+1,j-1)) + D. \tag{63}$$

Then, we have

$$\Phi_L^{i}(x^*(i,j)) \ge \Phi_L^{i}(x^*(i+1,j-1)) + D. \tag{64}$$

By (63) and (64), we obtain

$$\begin{aligned} \Phi_L^{i}(x^*(i+1,j-1)) - \Phi_L^{i+1}(x^*(i+1,j-1)) \\ \le \Phi_L^{i}(x^*(i,j)) - \Phi_L^{i+1}(x^*(i,j)), \end{aligned} \tag{65}$$

which is equivalent to (62) as mentioned above.

On the other hand, by the optimality of $\Phi_{\mathrm{OPT}}(1,i,j-1)$ and $\Phi_{\mathrm{OPT}}(1,i+1,j)$, we have

$$\Phi_{i,j-1}(x^*(i+1,j-1)) \ge \Phi_{i,j-1}(x^*(i,j-1)), \quad \text{and} \tag{66}$$

$$\Phi_{i+1,j}(x^*(i,j)) \ge \Phi_{i+1,j}(x^*(i+1,j)). \tag{67}$$

Then, by (62), (66) and (67), we obtain

$$\Phi_{\mathrm{OPT}}(1,i,j-1) - \Phi_{\mathrm{OPT}}(1,i+1,j-1) \le \Phi_{\mathrm{OPT}}(1,i,j) - \Phi_{\mathrm{OPT}}(1,i+1,j), \tag{68}$$

which implies that the lemma holds in Case 1. □

Theorem 15 [35] *The minisum k-facility location problem in a dynamic flow path network with uniform edge capacity can be solved in* $\min\{O(n^2\sqrt{k\log n} + n^2\log n), n^2 2^{O(\sqrt{\log k \log\log n})}\}$ *time.*

Results for the Case of General Edge Capacities We here introduce the latest results by [7] for the minisum k-facility location problem in a dynamic flow path network with general edge capacities. Although paper [7] has treated not only the case of general edge capacities but also that of uniform edge capacity and the time complexity of their algorithm for the case of uniform edge capacity has improved the previous one shown in Theorem 15, we do not refer to the details here because of space limitations.

Theorem 16 [7] *The minisum k-facility location problem in a dynamic flow path network with general edge capacities can be solved in* $O(kn \log^4 n)$ *time. The problem for the case of uniform edge capacity can be solved in* $O(kn \log^3 n)$ *time.*

4 Minimax Regret Facility Location Problems in Dynamic Flow Networks

The minimax regret facility location problems in dynamic flow networks have been studied in recent years. All of the previous studies on the problems have assumed that the edge capacity is uniform and facilities can be located at any point in the network. Cheng et al. [18] first studied the minimax regret 1-facility location problem in a dynamic flow path network adopting the maximum cost criterion and proposed an $O(n \log^2 n)$ time algorithm. This result was improved to $O(n \log n)$ by Higashikawa et al. [29] and Wang [44, 45] independently. Finally, Bhattacharya and Kameda [10] have developed an $O(n)$ time algorithm. They also studied the minimax regret 2-facility location problem in a dynamic flow path network adopting the maximum cost criterion and proposed an $O(n \log^4 n)$ time algorithm in [10]. The minimax regret k-facility location problem in a dynamic flow path network has first been studied by Arumugam et al. [3]. They developed two algorithms in [3]: the first one runs in $O(kn^2 \log^k n)$ time and the second one runs in $O(kn^3 \log n)$ time. The second result in [3] was improved to $O(kn^3)$ by Higashikawa [28]. For tree networks, Higashikawa et al. [32, 34] studied the minimax regret 1-facility location problem adopting the maximum cost criterion and proposed an $O(n^2 \log^2 n)$ time algorithm. Later, Bhattacharya and Kameda [10] developed an $O(n \log n)$ time algorithm, which is the best so far. Very recently, Golin and Sandeep [26] have studied the minimax regret k-facility location problem in a dynamic flow tree network adopting the maximum cost criterion and shown that the problem can be solved in $O(\max\{k^2, \log^2 n\} \cdot k^2 n^2 \log^5 n)$ time.

On the other hand, the problems adopting the total cost criterion have not been studied much except for the case of the 1-facility location in path networks. For the minimax regret 1-facility location problem in a dynamic flow path network adopting the total cost criterion, Higashikawa et al. [30, 31] provided an $O(n^3)$ time algorithm, and this result has been improved to $O(n^2 \log^2 n)$ recently by Bhattacharya et al. [9].

In the rest of this section, we introduce basic ideas for the reduction of scenarios to be considered in the minimax regret facility location problems in dynamic flow networks.

4.1 Dynamic Network Under Uncertain Supplies

A dynamic flow network under uncertain supplies $\mathcal{N} = (G = (V, E), W, l, c, \tau)$ consists of the same components as the one under fixed supplies (mentioned in Sect. 3.2.2) except for function w. Here, function W associates each vertex $v \in V$ with an interval of supply such that $W(v) = [w^-(v), w^+(v)]$ satisfying $0 < w^-(v) \leq w^+(v)$ instead of w. In a dynamic flow network under uncertain supplies, a particular assignment of supplies to vertices is called a scenario. Let $\mathcal{S}$ denote the Cartesian product of all $W(v)$ for $v \in V$, i.e., a set of scenarios:

$$\mathcal{S} = \prod_{v \in V} [w^-(v), w^+(v)]. \tag{69}$$

For a scenario $s \in \mathcal{S}$, the notation $w^s(v)$ is used to denote the supply of each vertex $v \in V$ under the scenario s.

4.2 Problems Adopting the Maximum Cost Criterion

For a location of facilities $\boldsymbol{x}$ and a scenario $s \in \mathcal{S}$, let $\Theta^s(\boldsymbol{x})$ denote the maximum cost of $\boldsymbol{x}$ under s. Also let $\boldsymbol{x}^s$ denote the minimax facility location under s. Given a location of facilities $\boldsymbol{x}$ and a scenario $s \in \mathcal{S}$, the regret adopting the maximum cost criterion of $\boldsymbol{x}$ under s is defined as follows:

$$RM^s(\boldsymbol{x}) = \Theta^s(\boldsymbol{x}) - \Theta^s(\boldsymbol{x}^s). \tag{70}$$

Then, given a facility location $\boldsymbol{x}$, the maximum regret adopting the maximum cost criterion of $\boldsymbol{x}$ is defined as follows:

$$RM_{\max}(\boldsymbol{x}) = \max\{RM^s(\boldsymbol{x}) \mid s \in \mathcal{S}\}. \tag{71}$$

Here, we treat a problem that requires finding $\boldsymbol{x}$ in a dynamic flow network minimizing $RM_{\max}(\boldsymbol{x})$. A scenario $s^* \in \mathcal{S}$ is called a worst case scenario adopting the maximum cost criterion for $\boldsymbol{x}$ if

$$s^* = \operatorname{argmax}\{RM^s(\boldsymbol{x}) \mid s \in \mathcal{S}\}. \tag{72}$$

4.2.1 *k*-Facility Location in Paths

The minimax regret k-facility location problems in a dynamic flow path network adopting the maximum cost criterion have been studied so far in [3, 10, 18, 28, 29, 44, 45]. As mentioned above, all these studies have assumed that the edge capacity is uniform and facilities can be located at any point in the network. Given a dynamic flow path network under uncertain supplies $\mathcal{N} = (P = (V, E), W, l, c', \tau)$ where P is the same as in Sect. 3.2.2 and c' is the uniform edge capacity, the problem requires finding a location of k facilities $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in P^k$ that minimizes $RM_{\max}(\boldsymbol{x})$.

Basic Properties for the 1-Facility Location The authors of [18, 29] first studied the case of $k = 1$ and proved several key properties that were useful even for the case of general k. When $k = 1$, we use the notation x and x^s to denote $\boldsymbol{x}$ and $\boldsymbol{x}^s$, respectively. If scenario s is fixed, function $\Theta^s(x)$ has the same properties as $\Theta_{1,n}(x)$ (refer to (5), (8)

and (9)), i.e., assuming that x is located on an edge e_{s_x} (not including endpoints) satisfying $1 \le s_x \le n-1$, $\Theta^s(\boldsymbol{x})$ is represented as

$$\Theta^s(x) = \max\{\Theta^s_L(x), \Theta^s_R(x)\}, \tag{73}$$

where

$$\Theta^s_L(x) = \max\left\{ \tau(x - v_l) + \frac{\sum_{1\le h\le l} w^s(v_h)}{c'} \;\middle|\; 1 \le l \le s_x \right\}, \tag{74}$$

$$\Theta^s_R(x) = \max\left\{ \tau(v_l - x) + \frac{\sum_{l\le h\le n} w^s(v_h)}{c'} \;\middle|\; s_x + 1 \le l \le n \right\}. \tag{75}$$

Similarly to $\Theta_{1,n}(x)$, $\Theta^s(x)$ is unimodal in x for a fixed s, and there uniquely exists x^s. By the definition of (70), $RM^s(x)$ is also unimodal in x for a fixed s, and then $RM_{\max}(x)$ is the upper envelope of unimodal functions by (71), which implies that $RM_{\max}(x)$ is unimodal in x.

Lemma 14 *$RM_{\max}(x)$ is unimodal in x.*

We next introduce the most important property in the 1-facility location, which was proved in [18, 29]. A main difficulty of the problem lies in evaluating $RM^s(x)$ over $s \in \mathscr{S}$ to compute $RM_{\max}(x)$ even for a fixed location $x \in P$ since the size of $\mathscr{S}$ is infinite. The authors of [18, 29] then proved that a worst case scenario for any $x \in P$ is included in a finite set of scenarios, called bipartite scenarios (in [18, 29], called dominant scenarios), defined as follows. A scenario s is said to be left-bipartite (resp. right-bipartite) if $w^s(v_j) = w^+(v_j)$ (resp. $w^-(v_j)$) over $j \in \{1, \dots, i\}$ and $w^s(v_j) = w^-(v_j)$ (resp. $w^+(v_j)$) over $j \in \{i+1, \dots, n\}$ for some $i \in \{1, \dots, n-1\}$. Let $\mathscr{S}_L$ (resp. $\mathscr{S}_R$) denote the set of all left-bipartite (resp. right-bipartite) scenarios. $\mathscr{S}_L$ consists of the following $n+1$ scenarios:

$$\begin{aligned} s^i_L &= (w^+(v_1), \dots, w^+(v_i), w^-(v_{i+1}), \dots, w^-(v_n)) \;\text{ for }\; 1 \le i \le n-1, \text{ and} \\ s^n_L &= (w^+(v_1), w^+(v_1), \dots, w^+(v_n)), \end{aligned} \tag{76}$$

and $\mathscr{S}_R$ consists of the following $n+1$ scenarios:

$$\begin{aligned} s^i_R &= (w^-(v_1), \dots, w^-(v_i), w^+(v_{i+1}), \dots, w^+(v_n)) \;\text{ for }\; 1 \le i \le n-1, \text{ and} \\ s^n_R &= (w^-(v_1), w^-(v_1), \dots, w^-(v_n)). \end{aligned} \tag{77}$$

It is clear that the number of such bipartite scenarios is $O(n)$.

Lemma 15 $|\mathscr{S}_L \cup \mathscr{S}_R| = O(n)$.

The authors of [18, 29] proved the following lemma.

Lemma 16 *Consider the minimax regret 1-facility location problems in a dynamic flow path network adopting the maximum cost criterion. For a location of a single facility $x \in P$, a worst case scenario belongs to $\mathscr{S}_L \cup \mathscr{S}_R$.*

Proof In the proof, we use the notation s_{i+} and s_{i-} for a given $s \in \mathscr{S}$ to denote scenarios such that

$$w^{s_{i+}}(v_i) = w^+(v_i) \text{ and } w^{s_{i+}}(v_j) = w^s(v_j) \text{ for } j \neq i, \text{ and}$$
$$w^{s_{i-}}(v_i) = w^-(v_i) \text{ and } w^{s_{i-}}(v_j) = w^s(v_j) \text{ for } j \neq i,$$

respectively. Let $s^* = \operatorname{argmax}\{RM^s(x) \mid s \in \mathscr{S}\}$. Here, we only prove for a facility location x such that $\Theta^{s^*}(x^{s^*})_L(x) \geq \Theta_R^{s^*}(x)$. Suppose that $v_{k-1} < x \leq v_k$ satisfying $2 \leq k \leq n$ and $l = \operatorname{argmax}\{\tau(x - v_l) + (\sum_{1 \leq h \leq l} w^{s^*}(v_h))/c' \mid 1 \leq l \leq k-1\}$, i.e.,

$$\Theta^{s^*}(x) = \tau(x - v_l) + \frac{\sum_{1 \leq h \leq l} w^{s^*}(v_h)}{c'}. \tag{78}$$

Then, we actually prove that $RM^{s^l_L}(x) \geq RM^{s^*}(x)$ holds. If s^* is not equal to s^l_L, we have two cases: [Case 1] there exists an integer i satisfying $1 \leq i \leq l$ such that $w^{s^*}(v_i) < w^+(v_i)$, and [Case 2] there exists an integer i satisfying $l+1 \leq i \leq n$ such that $w^{s^*}(v_i) > w^-(v_i)$. If we can show that $RM^{s^*_{i+}}(x) \geq RM^{s^*}(x)$ holds for Case 1 and $RM^{s^*_{i-}}(x) \geq RM^{s^*}(x)$ holds for Case 2, we will eventually obtain $RM^{s^l_L}(x) \geq RM^{s^*}(x)$ by repeatedly applying the same discussion as long as there exists such a vertex v_i.

[Case 1]: Let $\Delta = (w^+(v_i) - w^{s^*}(v_i))/c'$. We first notice

$$\Theta^{s^*_{i+}}(x) = \Theta_L^{s^*_{i+}}(x), \quad \text{and}$$
$$\Theta_L^{s^*_{i+}}(x) = \tau(x - v_l) + \frac{\sum_{1 \leq h \leq l} w^{s^*_{i+}}(v_h)}{c'} = \Theta^{s^*}(x) + \Delta$$

by (74) and (78). Thus, we have

$$\Theta^{s^*_{i+}}(x) = \Theta^{s^*}(x) + \Delta. \tag{79}$$

By the optimality of $x^{s^*_{i+}}$ under s^*_{i+}, $\Theta^{s^*_{i+}}(x^{s^*_{i+}}) \leq \Theta^{s^*_{i+}}(x^{s^*})$ holds. Here, we show

$$\Theta^{s^*_{i+}}(x^{s^*}) \leq \Theta^{s^*}(x^{s^*}) + \Delta \tag{80}$$

for the subcase of $x^{s^*} > v_i$ (the other case can be similarly treated). In this case, $\Theta_L^{s^*_{i+}}(x^{s^*}) \leq \Theta_L^{s^*}(x^{s^*}) + \Delta$ (see Fig. 6) and $\Theta_R^{s^*_{i+}}(x^{s^*}) = \Theta_R^{s^*}(x^{s^*})$ hold by the definitions of (74) and (75), which implies that (80) holds. Thus, we have

$$\Theta^{s^*_{i+}}(x^{s^*_{i+}}) \leq \Theta^{s^*}(x^{s^*}) + \Delta. \tag{81}$$

By (70), (79) and (81), we obtain $RM^{s^*_{i+}}(x) \geq RM^{s^*}(x)$.

[Case 2]: In this case, $\Theta^{s^*_{i-}}(x) = \Theta_L^{s^*_{i-}}(x)$ and $\Theta_L^{s^*_{i-}}(x) = \Theta_L^{s^*}(x)$ by (74) and (75). Thus, we have

$$\Theta^{s^*_{i-}}(x) = \Theta^{s^*}(x). \tag{82}$$

By the optimality of $x^{s^*_{i-}}$ under s^*_{i-}, $\Theta^{s^*_{i-}}(x^{s^*_{i-}}) \leq \Theta^{s^*_{i-}}(x^{s^*})$ holds. Also $\Theta^{s^*_{i-}}(x^{s^*}) \leq \Theta^{s^*}(x^{s^*})$ holds clearly. Thus, we have

$$\Theta^{s^*_{i-}}(x^{s^*_{i-}}) \leq \Theta^{s^*}(x^{s^*}). \tag{83}$$

By (70), (82) and (83), we obtain $RM^{s^*_{i-}}(x) \geq RM^{s^*}(x)$. □

We introduce another property which Higashikawa et al. [29] and Wang [44, 45] independently showed.

Lemma 17 *For a scenario $s \in \mathscr{S}$ and an integer i satisfying $1 \leq i \leq n$ such that $v_1 \leq v_i \leq x^s$ (resp. $x^s \leq v_i \leq v_n$), $v_i \leq x^{s_{i+}} \leq x^s$ (resp. $x^s \leq x^{s_{i+}} \leq v_i$) holds.*

Proof We only prove $v_i \leq x^{s_{i+}} \leq x^s$ for a given integer i satisfying $1 \leq i \leq n$ such that $v_1 \leq v_i \leq x^s$ (the other case can be similarly treated). We first prove $x^{s_{i+}} \leq x^s$ by contradiction: suppose that $x^{s_{i+}} > x^s$. Let x_{mid} be the midpoint of $x^{s_{i+}}$ and x^s:

$$x_{\mathrm{mid}} = \frac{x^{s_{i+}} + x^s}{2}.$$

We notice $x^s < x_{\mathrm{mid}} < x^{s_{i+}}$. Since an increasing function $\Theta_L^s(x)$ and a decreasing function $\Theta_R^s(x)$ intersect at $x = x^s$ and similarly $\Theta_L^{s_{i+}}(x)$ and $\Theta_R^{s_{i+}}(x)$ intersect at $x = x^{s_{i+}}$, we have

$$\Theta_R^s(x_{\mathrm{mid}}) < \Theta_L^s(x_{\mathrm{mid}}) \text{ and } \Theta_L^{s_{i+}}(x_{\mathrm{mid}}) < \Theta_R^{s_{i+}}(x_{\mathrm{mid}}). \tag{84}$$

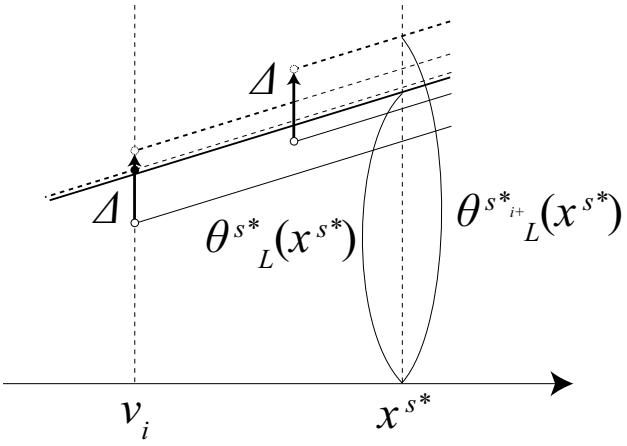

Fig. 6 Subcase of Case 1: $x^{s^*} > v_i$

Note that by $x^s < x_{\mathrm{mid}}$ and the assumption of $v_i \le x^s$, we have $v_i < x_{\mathrm{mid}}$. Therefore $\Theta^s_R(x_{\mathrm{mid}})$ does not change if $w^s(v_i)$ increases to $w^+(v_i)$, i.e.,

$$\Theta^{s_{i+}}_R(x_{\mathrm{mid}}) = \Theta^s_R(x_{\mathrm{mid}}). \tag{85}$$

By (84) and (85), we obtain $\Theta^{s_{i+}}_L(x_{\mathrm{mid}}) < \Theta^s_L(x_{\mathrm{mid}})$, which is a contradiction.

We next prove $v_i \le x^{s_{i+}}$ by contradiction: suppose that $x^{s_{i+}} < v_i$. First, by the definitions of (73), (74) and (75), we have

$$\Theta^{s_{i+}}_L(v_i) = \Theta^s_L(v_i) \text{ and } \Theta^{s_{i+}}_R(v_i) = \Theta^s_R(v_i), \tag{86}$$

i.e.,

$$\Theta^{s_{i+}}(v_i) = \Theta^s(v_i). \tag{87}$$

Note that $\Theta^{s_{i+}}_L(v_i) > \Theta^{s_{i+}}_R(v_i)$ when $x^{s_{i+}} < v_i$. By this and (86), we have $\Theta^s_L(v_i) > \Theta^s_R(v_i)$, which implies $v_i \ge x^s$. On the other hand, we have the condition of $v_i \le x^s$, that is,

$$x^s = v_i. \tag{88}$$

By the definitions of (73), (74) and (75), we also have

$$\Theta^s(x^{s_{i+}}) \le \Theta^{s_{i+}}(x^{s_{i+}}). \tag{89}$$

By the assumption of $x^{s_{i+}} \ne v_i$ and the optimality of $x^{s_{i+}}$ under s_{i+}, we have

$$\Theta^{s_{i+}}(x^{s_{i+}}) < \Theta^{s_{i+}}(v_i). \tag{90}$$

From (87), (88), (89) and (90), we can derive $\Theta^s(x^{s_{i+}}) < \Theta^s(x^s)$, which contradicts the optimality of x^s under s.

□

As a corollary of Lemma 17, we can see the following property.

Corollary 1 (i) *As long as* $x^{s^i_L} \ge v_{i+1}$ *holds,* $x^{s^i_L}$ *does not increase as* i *increases.* (ii) *Once* $x^{s^i_L} \le v_{i+1}$ *holds,* $x^{s^i_L}$ *never decreases as* i *increases.*

Sketch of Algorithms for the 1-Facility Location For the minimax regret 1-facility location problem in a dynamic flow path network with uniform edge capacity adopting the maximum cost criterion, Cheng et al. [18] developed the first algorithm based on Lemmas 14, 15 and 16, which is as follows. The algorithm by [18] first constructs a data structure in $O(n \log n)$ time so that for any integer i satisfying $1 \le i \le n$ and any bipartite scenario $s \in \mathscr{S}_L \cup \mathscr{S}_R$, $\Theta^s(v_i)$ can be computed in $O(\log n)$ time. Then, for a fixed bipartite scenario $s \in \mathscr{S}_L \cup \mathscr{S}_R$, the algorithm can compute $\Theta^s(x^s)$ in $O(\log^2 n)$ time by a binary search based on the unimodality of function $\Theta^s(x)$.

Thus, $\Theta^s(x^s)$ for all bipartite scenarios $s \in \mathscr{S}_L \cup \mathscr{S}_R$ can be computed in $O(n \log^2 n)$ time. After that, for a fixed i, the algorithm can compute $RM^s(v_i) = \Theta^s(v_i) - \Theta^s(x^s)$ (see (70)) for all bipartite scenarios $s \in \mathscr{S}_L \cup \mathscr{S}_R$ in $O(n \log n)$ time, which implies that by comparing all obtained $RM^s(v_i)$, $RM_{\max}(v_i)$ can be computed in $O(n \log n)$ time (see (71)). By the unimodality of function $RM_{\max}(x)$ mentioned in Lemma 14, the algorithm can apply a binary search to find $x \in P$ that minimizes $RM_{\max}(x)$. The authors of [18] thus developed an $O(n \log^2 n)$ time algorithm.

The algorithm by [18] was improved by Higashikawa et al. [29] and Wang [44, 45] independently. Although the algorithm by [29] constructs the same data structure as in [18], it computes $\Theta^s(x^s)$ for all bipartite scenarios $s \in \mathcal{S}_L \cup \mathscr{S}_R$ in $O(n \log n)$ time applying the monotonic property mentioned in Corollary 1, which contrasts with the algorithm by [18] computing the same values in $O(n \log^2 n)$ time as mentioned above. Also, the authors of [29] showed that for a fixed i, $RM^s(v_i)$ for all bipartite scenarios $s \in \mathscr{S}_L \cup \mathscr{S}_R$ can be computed in $O(n)$ time by the careful observation (in [18], it takes $O(n \log n)$ time as mentioned above). If the algorithm applies a binary search to find $x \in P$ that minimizes $RM_{\max}(x)$ as in [18], the problem can be solved in $O(n \log n)$ time in total. Bhattacharya and Kameda [10] have studied the same problem and developed an $O(n)$ time algorithm, which uses all of the properties mentioned in Lemmas 14, 15, 16 and Corollary 1 but does not construct the data structure as in [18, 29]. See [10] for more detail.

Theorem 17 [10] *The minimax regret 1-facility location problem in a dynamic flow path network with uniform edge capacity adopting the maximum cost criterion can be solved in $O(n)$ time.*

Basic Properties for the ***k****-Facility Location* The minimax regret k-facility location problem in a dynamic flow path network with uniform edge capacity adopting the maximum cost criterion was first studied by Arumugam et al. [3]. They assumed that all units of supply at a vertex are allocated the same evacuation path as in [33, 35]. Here, a scenario s is said to be tripartite if $w^s(v_h) = w^+(v_h)$ over $h \in \{i, \ldots, j\}$ and $w^s(v_h) = w^-(v_h)$ over $h \notin \{i, \ldots, j\}$ for some integers i, j satisfying $1 \le i \le j \le n$. Let $\mathscr{S}_T$ denote the set of all tripartite scenarios. It is clear that the number of such tripartite scenarios is $O(n^2)$.

Lemma 18 $|\mathscr{S}_T| = O(n^2)$.

The authors of [3] proved that a worst case scenario for any $\boldsymbol{x} \in P^k$ is a tripartite scenario based on Lemma 16.

Lemma 19 *Consider the minimax regret k-facility location problems in a dynamic flow path network adopting the maximum cost criterion. For a location of k facilities $\boldsymbol{x} \in P^k$, a worst case scenario belongs to $\mathscr{S}_T$.*

Sketch of Algorithms for the ***k****-Facility Location* The authors of [3] developed two algorithms: the first one runs in $O(kn^2 \log^k n)$ time and the second one runs in

$O(kn^3 \log n)$ time. Note that the second one is faster when $k > \log n/(\log \log n) + 1$, e.g., $k \geq 5$ for $n = 1000$. Based on Lemma 19, the second algorithm in [3] first computes the minimax k-facility location under every tripartite scenario, i.e., $\Theta^s(\boldsymbol{x}^s)$ for $s \in \mathscr{S}_T$, using an algorithm on the minimax k-facility location in a dynamic flow path network by [33]. The number of tripartite scenarios is $O(n^2)$ by Lemma 18, and the computation of $\Theta^s(\boldsymbol{x}^s)$ for each $s \in \mathscr{S}_T$ takes $O(kn \log n)$ time using the algorithm by [33], thus computing $\Theta^s(\boldsymbol{x}^s)$ for all $s \in \mathscr{S}_T$ takes $O(kn^3 \log n)$ in total. The authors of [3] also showed that the other part can be done in $O(n^3)$ time, which implies that computing $\Theta^s(\boldsymbol{x}^s)$ for all $s \in \mathcal{S}_T$ dominates the other part in the sense of time complexity. The time complexity of this computation has been improved to $O(kn^3)$ by [28] as the algorithm by [33] has been improved to an $O(kn)$ time algorithm by [35], which implies that the minimax regret k-facility location problem in a dynamic flow path network can be solved in $O(kn^3)$ time.

Theorem 18 [28] *The minimax regret k-facility location problem in a dynamic flow path network with uniform edge capacity adopting the maximum cost criterion can be solved in $O(kn^3)$ time.*

4.2.2 *k*-Facility Location in Trees

The minimax regret 1-facility location problems in a dynamic flow tree network adopting the maximum cost criterion have been studied in [10, 32, 34] so far. Very recently, Golin and Sandeep [26] have studied the case of k-facility location.

In this section, we mainly introduce the algorithms for the 1-facility location in [32, 34] and mention the results for the k-facility location in [26] because of space limitations. Given a dynamic flow tree network under uncertain supplies $\mathcal{N} = (T = (V, E), W, l, c', \tau)$ where T is the same as in Sect. 3.2.3 and c' is the uniform edge capacity, the problem requires finding a location of a single facility $x \in T$ that minimizes $RM_{\max}(x)$. Here, the notation T is abused to denote the set of all points on T. In the following, since we consider the 1-facility location, we use the notation x and x^s to denote $\boldsymbol{x}$ and $\boldsymbol{x}^s$, respectively.

Basic Properties for the 1-Facility Location The authors of [32, 34] proved several properties in the minimax regret 1-facility location problem in a dynamic flow tree network adopting the maximum cost criterion. As in Sect. 3.2.3, we here use the same notation $d(x, y)$ for two points $x, y \in T$, $\delta(x)$ for a point $x \in T$, and $T(x, v)$ a point $x \in T$ and a vertex $v \in V$. Also given a scenario $s \in \mathscr{S}$, for a point $x \in T$ and a vertex $u \in \delta(x)$, let $\Theta^s(x, u)$ denote the maximum cost of x for all supplies on $T(x, u)$ under s. Then, referring to (40) and (42), $\Theta^s(x)$ is represented (in the fractional flow model) as follows:

$$\Theta^s(x) = \max \left\{ \tau d(x, v_i) + \frac{\sum_{i \leq j \leq n'} w^s(v_j)}{c'} \;\middle|\; 1 \leq i \leq n' \right\}, \tag{91}$$

where $v_1 = \text{argmax}\{\Theta^s(x,u) \mid u \in \delta(x)\}$ and there are n' vertices in $T(x, v_1)$ named $v_1, v_2, \ldots, v_{n'}$ such that $d(x, v_j) \leq d(x, v_{j+1})$ for $1 \leq j \leq n' - 1$.

First, we confirm the following two lemmas.

Lemma 20 *For a scenario $s \in \mathscr{S}$, along a path from a leaf to another leaf in T, $\Theta^s(x)$ is unimodal in x.*

Lemma 21 *For a vertex $v \in V$ and a scenario $s \in \mathscr{S}$, if $u^* = \text{argmax}\{\Theta^s(v,u) \mid u \in \delta(v)\}$ holds, there exists $x^s \in T(V(v,u^*) \cup \{v\})$.*

These immediately follow by Lemmas 8 and 9, respectively. By (70), (71) and Lemma 20, we can also see the following lemma.

Lemma 22 *Along a path from a leaf to another leaf in T, $RM_{\max}(x)$ is unimodal in x.*

We here introduce a concept of dominant scenarios for a vertex $v \in V$. Suppose that $u \in \delta(v)$, n' is the number of vertices in $T(v,u)$ and $v_1(=u), v_2, \ldots, v_{n'}$ are vertices in $T(v,u)$ such that $d(v,v_i) \leq d(v,v_{i+1})$ for $1 \leq i \leq n'-1$. We now consider a scenario $s \in \mathscr{S}$ such that $w^s(v_i) = w^+(v_i)$ for $v_i \in T(v,u)$ such that $l \leq i \leq n'$ for some l with $1 \leq l \leq n'$ and $w^s(v') = w^-(v')$ for all the other vertices $v' \in V$. In the following, such a scenario is said to be dominant for v, and represented by $s(v,v_l)$. Then, let $\mathscr{S}_D(v,u) = \{s(v,v_l) \mid 1 \leq l \leq n'\}$, and also let $\mathscr{S}_D(v) = \bigcup_{u \in \delta(v)} \mathscr{S}_D(v,u)$.

Note that $\mathscr{S}_D(v)$ consists of $n-1$ scenarios.

Lemma 23 *For a vertex $v \in V$, $|\mathscr{S}_D(v)| = O(n)$.*

The authors of [32, 34] proved the following lemma, which follows by (91) and Lemma 16.

Lemma 24 *Consider the minimax regret 1-facility location problems in a dynamic flow tree network adopting the maximum cost criterion. When a facility x is located at a vertex $v \in V$, a worst case scenario for x belongs to $\mathscr{S}_D(v)$.*

Also, letting x^* denote a point x that minimizes $RM_{\max}(x)$ for $x \in T$, the authors of [32, 34] proved the following lemma.

Lemma 25 *For a vertex $v \in V$, if $s^* = \text{argmax}\{RM^s(v) \mid s \in \mathscr{S}\}$ and $u^* = \text{argmax}\{\Theta^{s^*}(v,u) \mid u \in \delta(v)\}$ hold, there exists $x^* \in T(V(v,u^*) \cup \{v\})$.*

Proof We prove by contradiction: suppose that there exists $x^* \in T(v,u)$ or on an edge vu (not including endpoints) for some $u \in \delta(v)$ with $u \neq u^*$. By Lemma 21, there exists $x^{s^*} \in T(V(v,u^*) \cup \{v\})$. Now, let us consider a path which goes through x^{s^*}, v and x^* in this order. By Lemma 20, $\Theta^{s^*}(x)$ is increasing in x when x moves along the path from x^{s^*} to x^*, which implies that $\Theta^{s^*}(x^*) > \Theta^{s^*}(v)$ holds. Thus, $RM^{s^*}(x^*) > RM^{s^*}(v)$ also holds by (70).

We have $RM_{\max}(x^*) \geq RM^{s^*}(x^*)$ by the maximality of $RM_{\max}(x^*)$ and $RM^{s^*}(v) = RM_{\max}(v)$ by definition of s^*, thus $RM_{\max}(x^*) > RM_{\max}(v)$ holds, which contradicts the optimality of x^*. □

Sketch of an Algorithm for the 1-Facility Location The authors of [32, 34] developed an algorithm that computes $x^* \in T$ based on the above mentioned properties. For ease of explanation, we here assume that a facility is located on a vertex although the authors of [32, 34] allowed that a facility can be located at any point on a tree. We first explain how an algorithm by [32, 34] computes $RM_{\max}(v)$ for a vertex $v \in V$. Given a dominant scenario $s \in \mathscr{S}_D(v)$, by Theorem 12, $\boldsymbol{\Theta}^s(v)$ and $\boldsymbol{\Theta}^s(x^s)$ can be computed in $O(n \log n)$ time, respectively. This implies that $RM^s(v)$ can be computed in $O(n \log n)$ time (see (70)). By Lemmas 23 and 24, we only need to consider $O(n)$ dominant scenarios for a particular v. Thus, $RM_{\max}(v)$ can be computed in $O(n^2 \log n)$ time (see (71)). We assume that when $RM_{\max}(v)$ is computed, $s^* = \text{argmax}\{RM^s(v) \mid s \in \mathscr{S}\}$ and $u^* = \text{argmax}\{\boldsymbol{\Theta}^{s^*}(v,u) \mid u \in \delta(v)\}$ are also computed.

To find $x^* \in T$, we can apply a similar approach to the one presented by [28, 32, 34] (mentioned in Sect. 3.2.3) that finds the minimax 1-facility location in a dynamic flow tree network with uniform edge capacity since Lemma 25 holds. Therefore the algorithm can find $x^* \in T$ by computing $RM_{\max}(v)$ for $O(\log n)$ times, which implies that x^* can be computed in $O(n^2 \log^2 n)$ time. The authors of [32, 34] thus developed an $O(n^2 \log^2 n)$ time algorithm. Bhattacharya and Kameda [10] have developed an $O(n \log n)$ time algorithm using a better data structure. See [10] for more detail.

Theorem 19 [10] *The minimax regret 1-facility location problem in a dynamic flow tree network with uniform edge capacity adopting the maximum cost criterion can be solved in $O(n \log n)$ time.*

Results for the k-Facility Location We here introduce the latest results in [26] for the minimax regret k-facility location problem in a dynamic flow tree network with uniform edge capacity adopting the maximum cost criterion

Theorem 20 [26] *The minimax regret k-facility location problem in a dynamic flow tree network with uniform edge capacity adopting the maximum cost criterion can be solved in $O(\max\{k^2, \log^2 n\}k^2 n^2 \log^5 n)$ time.*

4.3 Problems Adopting the Total Cost Criterion

For a location of facilities $\boldsymbol{x}$ and a scenario $s \in \mathcal{S}$, let $\boldsymbol{\Phi}^s(\boldsymbol{x})$ denote the maxisum cost of $\boldsymbol{x}$ under s. Also let $\boldsymbol{x}^s$ denote the minisum facility location under s. Given a location of facilities $\boldsymbol{x}$ and a scenario $s \in \mathscr{S}$, the regret adopting the total cost criterion of $\boldsymbol{x}$ under s is defined as follows:

$$RT^s(\boldsymbol{x}) = \boldsymbol{\Phi}^s(\boldsymbol{x}) - \boldsymbol{\Phi}^s(\boldsymbol{x}^s). \tag{92}$$

Then, given a facility location $\boldsymbol{x}$, the maximum regret adopting the total cost criterion of $\boldsymbol{x}$ is defined as follows:

$$RT_{\max}(\boldsymbol{x}) = \max\{RT^s(\boldsymbol{x}) \mid s \in \mathscr{S}\}. \tag{93}$$

Here, we treat a problem that requires finding $\boldsymbol{x}$ in a dynamic flow network minimizing $RT_{\max}(\boldsymbol{x})$. A scenario $s^* \in \mathscr{S}$ is called a worst case scenario adopting the total cost criterion for $\boldsymbol{x}$ if

$$s^* = \operatorname{argmax}\{RT^s(\boldsymbol{x}) \mid s \in \mathscr{S}\}. \tag{94}$$

4.3.1 *k*-Facility Location in Paths

The minimax regret 1-facility location problem in a dynamic flow path network adopting the total cost criterion has been studied in [9, 30, 31]. As mentioned above, this study has also assumed that the edge capacity is uniform and a facility can be located at any point in the network. Given a dynamic flow path network $\mathscr{N} = (P = (V, E), w, l, c, \tau)$ defined in Sect. 4.3.1, the problem requires finding a location of a single facility $x \in P$ that minimizes $RT_{\max}(x)$. For the problem, the authors of [30, 31] proposed an $O(n^3)$ time algorithm. Very recently, Bhattacharya et al. [9] have improved the time complexity in [30, 31] to $O(n^2 \log^2 n)$. We do not refer to the details here because of space limitations.

Theorem 21 [9] *The minimax regret 1-facility location problem in a dynamic flow path network with uniform edge capacity adopting the total cost criterion can be solved in* $O(n^2 \log^2 n)$ *time.*

5 Conclusion

In this paper, we surveyed recent developments of algorithms for facility location problems in dynamic flow networks that were motivated by evacuation planning problems.

Contrasted with classical 1-center and 1-median problems, we showed the difficulty and approximability of solving 1-facility location problems for general networks although NP-hardness has not be proven yet, which is currently open. We then showed polynomial time algorithms for the problems in path and tree networks. We also showed the results for minimax regret versions of the problems where the weights (the number of evacuees) on vertices are not fixed but the only interval where the weight exists is known for every vertex.

As we showed, there are still many open problems. To attack those problems, we believe that new ideas and techniques are required. In this sense, we hope that many of the readers will have an interest in this problem.

There is a 70–80% percent chance that the Nankai Trough Earthquake will occur within the coming 30 years [1]. In Japan, there are many small towns on the coastal area facing the Pacific Ocean whose local governments are faced with the serious problem that they have to spend most of their budget to build tsunami evacuation

buildings to reduce the loss of human lives from a tsunami triggered by the earthquake. In this respect, we hope that the methods developed for facility location problems will help to reduce the budget used for such disaster prevention and help reduce the loss of human lives.

Funding Both of two authors are Supported by JST CREST (JPMJCR1402) and JSPS KAKENHI Grant-in-Aid for Scientific Research (B) (JP19H04068). In addition, Yuya HIgashikawa is Supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) (JP17K12641).

References

1. http://www.mlit.go.jp/river/mlit_at_wcdrr/pdf/Forum2_en.pdf
2. Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Upper Saddle River: Prentice Hall.
3. Arumugam, G. P., Augustine, J., Golin, M. J., & Srikanthan, P. (2014). A polynomial time algorithm for minimax-regret evacuation on a dynamic flow path. *CoRR* arXiv:1404.5448.
4. Averbakh, I., & Berman, O. (2000). Algorithms for therobust 1-center problem on a tree. *European Journal of Operational Research*, *123*(2), 292–302.
5. Belmonte, R., Higashikawa, Y., Katoh, N., & Okamoto, Y. (2015). Polynomial-time approximability of the k-Sink Location problem. *CoRR* arXiv:1503.02835.
6. Benkoczi, R., Bhattacharya, B., Higashikawa, Y., Kameda, T., & Katoh, N. (2018). Minsum k-sink problem on dynamic flow path networks. In *Proc. the 29th international workshop on combinatorial algorithms* (IWOCA 2018), LNCS **10979**, pp. 78–89.
7. Benkoczi, R., Bhattacharya, B., Higashikawa, Y., Kameda, T., & Katoh, N. Minsum k-sink problem on path networks. *Theoretical Computer Science* **(accepted)**.
8. Bhattacharya, B., Golin, M. J., Higashikawa, Y., Kameda, T., & Katoh, N. (2017). Improved algorithms for computing k-sink on dynamic flow path networks. In *Proc. the 15th algorithms and data structures symposium* (WADS 2017), LNCS **10389**, pp. 133–144.
9. Bhattacharya, B., Higashikawa, Y., Kameda, T., & Katoh, N. (2018). An $O(n^2 \log^2 n)$ time algorithm for minmax regret minsum sink on path networks. In *Proc. the 29th international symposium on algorithms and computation* (ISAAC 2018), LIPIcs **123**, pp. 14:1–14:13.
10. Bhattacharya, B., & Kameda, T. (2015). Improved algorithms for computing minmax regret sinks on dynamic flow path and tree networks. *Theoretical Computer Science*, *607*, 411–425.
11. Bhattacharya, B., Kameda, T., & Song, Z. (2013). A linear time algorithm for computing minmax regret 1-median on a tree network. *Algorithmica*, *70*(1), 1–20.
12. Brodal, G. S., Georgiadis, L., & Katriel, I. (2008). An $O(n \log n)$ version of the Averbakh-Berman algorithm for the robust median of a tree. *Operations Research Letters*, *36*(1), 14–18.
13. Burkard, R. E., Dlaska, K., & Klinz, B. (1993). The quickest flow problem. *Mathematical Methods of Operations Research*, *37*(1), 31–58.
14. Chen, D., & Golin, M. J. (2016). Sink evacuation on trees with dynamic confluent flows. In *Proc. the 27th international symposium on algorithms and computation* (ISAAC 2016), LIPIcs **64**, pp. 25:1–25:13.
15. Chen, D., & Golin, M. J. (2018). Minmax centered k-partitioning of trees and applications to sink evacuation with dynamic confluent flows. *CoRR* arXiv:1803.09289.
16. Charikar, M., Guha, S., Tardos, É., & Shmoys, D. B. (2002). A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, *65*(1), 129–149.

17. Chen, B., & Lin, C. (1998). Minmax-regret robust 1-median location on a tree. *Networks*, *31*(2), 93–103.
18. Cheng, S. W., Higashikawa, Y., Katoh, N., Ni, G., Su, B., & Xu, Y. (2013). Minimax regret 1-sink location problems in a dynamic flow path network. In *Proc. the 10th annual conference on theory and applications of models of computation* (TAMC 2013), LNCS **7876**, pp. 121–132.
19. Conde, E. (2008). A note on the minmax regret centdian location on trees. *Operations Research Letters*, *36*(2), 271–275.
20. Daskin, M. S. (2013). *Network and discrete location: Models, algorithms, and applications*. Oxford: Wiley.
21. Dyer, M. E., & Frieze, A. M. (1985). A simple heuristic for the *p*-center problem. *Operations Research Letters*, *3*(6), 285–288.
22. Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. Berlin: Springer.
23. Ford, L. R, Jr., & Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, *6*, 419–433.
24. Ford, L. R, Jr., & Fulkerson, D. R. (1962). *Flows in networks*. Princeton: Princeton University Press.
25. Frederickson, G. N., & Johnson, D. B. (1983). Finding *k*th paths and *p*-centers by generating and searching good data structures. *Journal of Algorithms*, *4*(1), 61–80.
26. Golin, M. J., & Sandeep, S. (2018). Minmax-regret *k*-sink location on a dynamic flow tree network with uniform capacities. *CoRR* arXiv:1806.03814.
27. Hall, A., Hippler, S., & Skutella, M. (2007). Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, *379*(3), 387–404.
28. Higashikawa, Y. (2014). Studies on the space exploration and the sink location under incomplete information towards applications to evacuation planning. *Doctoral Dissertation accepted by Kyoto University*.
29. Higashikawa, Y., Augustine, J., Cheng, S. W., Golin, M. J., Katoh, N., Ni, G., et al. (2015). Minimax regret 1-sink location problem in a dynamic flow path network. *Theoretical Computer Science*, *588*, 24–36.
30. Higashikawa, Y., Cheng, S. W., Kameda, T., Katoh, N., & Saburi, S. (2016). Minimax regret 1-median problem in a dynamic flow path network. In *Proc. the 27th international workshop on combinatorial algorithms* (IWOCA 2016), LNCS **9843**, pp. 122–134.
31. Higashikawa, Y., Cheng, S. W., Kameda, T., Katoh, N., & Saburi, S. (2018). Minimax regret 1-median problem in a dynamic flow path network. *Theory of Computing Systems*, *62*(6), 1392–1408.
32. Higashikawa, Y., Golin, M. J., & Katoh, N. (2014). Minimax regret sink location problem in a dynamic flow tree network with uniform capacity. In *Proc. the 8th international workshop on algorithms and computation* (WALCOM 2014), LNCS **8344**, pp. 125–137.
33. Higashikawa, Y., Golin, M. J., & Katoh, N. (2014). Multiple sink location problems in a dynamic flow path network. In *Proc. the 10th international conference on algorithmic aspects of information and management* (AAIM 2014), LNCS **8546**, pp. 149–161.
34. Higashikawa, Y., Golin, M. J., & Katoh, N. (2014). Minimax regret sink location problem in a dynamic flow tree network with uniform capacity. *Journal of Graph Algorithms and Applications*, *18*(4), 539–555.
35. Higashikawa, Y., Golin, M. J., & Katoh, N. (2015). Multiple sink location problems in a dynamic flow path network. *Theoretical Computer Science*, *607*, 2–15.
36. Hochbaum, D. S., & Shmoys, D. B. (1985). A best possible approximation algorithm for the *k*-center problem. *Mathematics of Operations Research*, *10*(2), 180–184.
37. Hoppe, B., & Tardos, É. (2000). The quickest transshipment problem. *Mathematics of Operations Research*, *25*(1), 36–62.
38. Kamiyama, N., Katoh, N., & Takizawa, A. (2006). An efficient algorithm for evacuation problem in dynamic flow network flows with uniform arc capacity. *IEICE Transactions*, *89–D(8)*, 2372–2379.
39. Kamiyama, N., Katoh, N., & Takizawa, A. (2009). An efficient algorithm for the evacuation problem in a certain class of networks with uniform path-lengths. *Discrete Applied Mathematics*, *157*(17), 3656–3664.
40. Kang, A. N. C., & Ault, D. A. (1975). Some properties of a centroid of a free tree. *Information Processing Letters*, *4*(1), 18–20.

41. Mamada, S., Uno, T., Makino, K., & Fujishige, S. (2006). An $O(n \log^2 n)$ algorithm for the optimal sink location problem in a dynamic flow tree network. *Discrete Applied Mathematics*, *154*(16), 2387–2401.
42. Niedermeier, R. (2006). *Invitation to Fixed-Parameter Algorithms*. Oxford: Oxford University Press.
43. Schieber, B. (1998). Computing a minimum weight k-link path in graphs with the concave monge property. *Journal of Algorithms*, *29*(2), 204–222.
44. Wang, H. Minmax regret 1-facility location on uncertain path networks. In *Proc. the 24th international symposium on algorithms and computation* (ISAAC 2013), pp. 733–743.
45. Wang, H. (2014). Minmax regret 1-facility location on uncertain path networks. *European Journal of Operational Research*, *239*(3), 636–643.