H. N. Nath, U. Pyakurel, T. N. Dhamala, and S. Dempe
Dynamic Network Flow Location Models and Algorithms for Quickest Evacuation Planning

PREPRINT 2018-12
Fakultät für Mathematik und Informatik
H. N. Nath, U. Pyakurel, T. N. Dhamala, and S. Dempe

Dynamic Network Flow Location Models and Algorithms for Quickest Evacuation Planning

TU Bergakademie Freiberg
Fakultät für Mathematik und Informatik
Prüferstraße 9
09599 FREIBERG
http://tu-freiberg.de/fakult1
DYNAMIC NETWORK FLOW LOCATION MODELS AND ALGORITHMS FOR QUICKEST EVACUATION PLANNING

HARI N. NATH\textsuperscript{1}, URMILA PYAKUREL\textsuperscript{2}, TANKA N. DHAMALA\textsuperscript{3} STEPHAN DEMPE\textsuperscript{4}

Abstract. Dynamic network flow problems have wide applications in evacuation planning. From a given subset of arcs in a directed network, choosing the suitable arcs for facility location is very important in the optimization of flows in emergency cases. Because of the decrease in the capacity of an arc by placing a facility in it, there may be a reduction in the maximum flow or increase in the quickest time. In this work, we consider a problem of identifying the optimal facility locations so that the increase in the quickest time is minimum. Introducing the quickest FlowLoc problem, we give strongly polynomial time algorithms to solve the single facility case. Realizing NP-hardness of the multi-facility case, we develop a mixed integer programming formulation of it and give a polynomial time heuristic for its solution. Because of the growing concerns of arc reversals in evacuation planning, we introduce quickest ContraFlowLoc problem and present exact algorithms to solve single-facility case and a heuristic to solve the multi-facility case, both of which have polynomial time complexity. The solutions thus obtained here are practically important, particularly in evacuation planning, to systematize traffic flow with facility allocation minimizing evacuation time.

1. Introduction

The choice of locations for the facilities such as hospitals, warehouses, stores, fire-brigades, security offices, etc. plays an important role in normal as well as in emergency disastrous situations. As in the normal situations, the mathematical models used to make location decisions in emergency situations are: (a) covering models which locate the optimal locations to cover all demand points or maximal number of demand points, (b) $P$-median models to determine $P$ locations to minimize the average (or total) distance between demand points and facilities, (c) $P$-Center models to minimize the maximum distance between any demand point and its nearest facility. For example, in Large Scale Emergency Medical Service Facility Location Model (LEMS) presented in Jia et al. [13], there is a use of the aforementioned models.

\textsuperscript{1}DYNAMIC NETWORK FLOW LOCATION MODELS AND ALGORITHMS FOR QUICKEST EVACUATION PLANNING

1991 Mathematics Subject Classification. 2010 Mathematics Subject Classification. Primary: 90B10, 90C27, 68Q25; Secondary: 90B06, 90B20.

Key words and phrases. Network flow; facility location; evacuation planning; quickest flow; contraflow.

The research has been carried out under the AvH Research Group Linkage Program between TU Bergakademie Freiberg, Germany and Central Department of Mathematics, TU Kathmandu, Nepal. The first author acknowledges the support of UGC Nepal and TU Bergakademie Freiberg. The second author acknowledges the AvH for the George Foster Fellowship for Post-doctoral Researchers.

1
Evacuation planning is an integral part of disaster management. Recently, there is a growing trend of incorporating location decisions in evacuation planning. The following are some examples.

(i) Pick-up location models: An et al. [2] formulate a model to determine the optimal pick-up location, evacuee-to-facility assignment priorities, evacuation service rates that minimizes the total expected system cost. In integrated bus evacuation problem, Goerigk et al. [10] choose pick-up locations to minimize the maximum travel time over all buses. Kulshrestha et al. [17] use robust optimization to locate pick-up locations when number of evacuees is uncertain.

(ii) Rescue Transfer Location Models: An et al. [2] formulate a model to locate rescue transfer locations, where rescue team departs from the rescue center, rides a vehicle to rescue transfer center, and walks to each rescue group to provide aid with an objective to minimize the total expected travel cost.

(iii) Shelter Location Models: Sherali et al. [36] formulate a location allocation model to minimize the total vehicle hours, and a discrete median location model to locate shelters for evacuation, while Kongsomsaksakul et al. [15] use bi-level programming approach to determine shelter locations, in which the upper level determines the shelter locations to minimize the total evacuation time and the lower level is formulated as combined trip distribution and assignment problem. Ng et al. [19] also use the same approach in which the lower level is deterministic user equilibrium model as described by Sheffi [37]. In integrated bus evacuation problem, Goerigk et al. [10] choose shelters to minimize the maximum travel time over all buses while in comprehensive evacuation planning, Goerigk et al. [9] formulate a multiple commodity, multi-criteria problem to minimize total evacuation time, risk exposure of evacuees, and number of shelters that are used.

(iv) Flow Location (FlowLoc) Models: Optimizing traffic flow is a very important aspect of evacuation planning. The common methods to optimize traffic flow are traffic simulation, models based on fluid dynamics, control theory, variational inequalities, and network flow. Since simulation does not explicitly allow for optimization, and models based on differential equations are not capable of handling large networks, network flow approach has been the most appropriate way of modeling traffic flow (Köhler and Skutella [16]). For details on network flow approach for evacuation planning problems, we refer to Dhamala et al. [5], Rupp [33], Hamacher et al. [11], Heller and Hamacher [12] combine location decisions with flow decisions in a network flow problem observing that the placement of a facility on an arc of a network may result into a reduction in the maximum flow value. Given a set of facilities and set of arcs on which facilities are to be placed, their approach is to find an allocation of the facilities to the arcs so that the reduction in the maximum flow is minimum.

The main aim of optimizing traffic flow in an emergency evacuation process is to maximize flow and/or to minimize the evacuation time using the road network. In such situations, people are discouraged to go towards risk areas from safer places. As a result the road segments heading towards the safe areas become overly congested and those heading towards the risk areas become empty. To maximize the flow and to minimize the evacuation time, in such situations, converting a two-way road segment to one-way in an appropriate direction becomes advantageous. This
is known as contraflow configuration, which reverses the direction of the traffic on empty road segments towards the sinks so that the capacity of the road segment is increased. Contraflow configuration not only increases flow value but also reduces the traffic jam and makes the traffic smooth. But to identify appropriate direction of the arcs of a network to maximize the flow is a difficult optimization problem, known as a contraflow problem. Different heuristic techniques to solve the contraflow problem in which at least 40% evacuation time can be reduced by reverting at most 30% arcs can be found in Kim et al. [14].

Apart from heuristic techniques, recent research also focuses on analytical techniques to find exact solution of contraflow problem after Rebennack et al. [32] introduced algorithms to solve the single-source single-sink maximum contraflow and quickest contraflow problems optimally in polynomial time. The earliest arrival and the maximum contraflow problems are solved with the temporally repeated flow solutions in Dhamala and Pyakurel [4, 22] with discrete time setting. The solution procedures for such problems in continuous time setting are described in Pyakurel and Dhamala [24]. In Pyakurel and Dhamala [23], authors design algorithms to solve the earliest arrival contraflow on single-source single-sink network in pseudo-polynomial time. They also introduce the lex-maximum dynamic contraflow problem in which flow is maximized in given priority ordering and construct solution algorithms with polynomial time complexity. Algorithms to these problems in continuous time by using nice property of natural transformation can be found in Pyakurel and Dhamala [24, 26]. With given supplies and demands, the earliest arrival transshipment contraflow problem is modeled in discrete time and solved it on multi-source network with polynomial time algorithm in Pyakurel and Dhamala [25]. With zero transit time on each arc, the problem is also solved on multi-sink network in polynomial time complexity. For the multi terminal network, they present approximation algorithms to solve the earliest arrival transshipment contraflow problem. The discrete solutions are extended into continuous time in Pyakurel and Dhamala [24, 26]; Pyakurel et al. [27]. The maximum dynamic and earliest arrival contraflow problems are generalized in Pyakurel et al. [31].

Moreover, the first temporally repeated flow algorithm to solve the quickest contraflow problem has been presented by Pyakurel et al. [28]. Considering a case of Kathmandu road network, the comparison of the quickest time before and after contraflow configuration show that a significant decrease in the quickest time can be attributed to contraflow configuration and the decrease in quickest time increases with the number of evacuee-vehicles. They also present an approximate algorithm to solve the quickest contraflow problem with load dependent transit time on each arc.

The analytical techniques discussed above use arc-based formulation of network flow problems. Recently path-based formulation of the similar problem with abstract flow on abstract networks is also gaining attention. Pyakurel et al. [27] introduce contraflow technique in abstract networks, present algorithms to solve maximum static and maximum dynamic contraflow problems with continuous time setting and realize that if the minimum dynamic cut capacities on two terminal network are symmetric, then the flow value can be increased up to double with partial contraflow reconfiguration. The models and algorithms for the abstract contraflow problems with discrete time setting have been investigated in Dhungana et al. [7]. With a view to save unused capacities of arcs during evacuation process, Pyakurel...
et al. [29, 30] investigate the partial contraflow problem and present algorithms to solve various problems related to abstract flow.

Motivated by the work of Hamacher et al. [11], our main focus in this paper is to introduce flow location models and develop efficient solution procedures to identify the allocation of facilities on arcs, with and without contraflow configuration, so that the increase in the quickest time is minimum. To facilitate the evacuation process, placing facilities on the road segments obstruct traffic flow resulting in the increase of the evacuation time. From a given set of road segments for the facilities to be placed, our approach is to choose those which have minimum impact on the increase in the transportation time. This is important in evacuation planning, particularly, when the given number of evacuees are to be transferred to the safe destinations as quickly as possible.

The paper is organized as follows. The basic terminology, notations and flow models necessary to the paper is considered in Section 2. Section 3 investigates the quickest FlowLoc problem and presents strongly polynomial algorithms to solve single facility cases and polynomial heuristic to solve multiple facility case. In Section 4, we combine contraflow with location decision on arcs and Section 5 concludes the paper with further research directions.

2. Basic concepts

In this section, we give some basic ideas used in this paper in an attempt to make it self-contained. We represent a transportation network by a directed graph in which the intersections of roads (or some other points on a road, if needed) denote the nodes and the road segments between any two nodes represent arcs. The direction of the traffic flow in a road segment is the direction of the corresponding arc. Something that moves from one node to the other via arcs is known as a flow.

We represent a directed network (also known as evacuation network in this paper) with the set of nodes $V$, set of arcs $A$, capacity $b : A \to \mathbb{R}_{\geq 0}$, travel time $\tau : A \to \mathbb{R}_{\geq 0}$, the source node $s \in V$ and the sink node $d \in V$ by $N = (V, A, b, \tau, s, d)$. The capacity of an arc limits the flow rate on the arc and travel time represents the time the flow takes to travel on the arc. We denote the number of nodes $|V|$ by $n$, the number of arcs $|A|$ by $m$, the set of incoming arcs to the node $i$ by $A_{i}^{\text{in}}$ and the set of arcs going out of it by $A_{i}^{\text{out}}$ i.e.

$$A_{i}^{\text{in}} = \{ e \in A : e = (j, i) \text{ for some } j \in V \}$$
$$A_{i}^{\text{out}} = \{ e \in A : e = (i, j) \text{ for some } j \in V \}$$

2.1. Static and dynamic flows. Let $x : A \to \mathbb{R}_{\geq 0}$ be a function of non-negative values, where $x(e)$ or $x_e$ is considered as the flow value on $e \in A$. For each $i \in V$, we denote the excess of $x$ at $i \in V$ by

$$\text{exc}_{x}(i) = \sum_{e \in A_{i}^{\text{in}}} x(e) - \sum_{e \in A_{i}^{\text{out}}} x(e)$$

which is the difference of the flow value entering $i$ and that going out from $i$.

**Definition 2.1.** For two distinct nodes $s, d \in V$, $x$ is called a static $s$-$d$ flow if

$$\text{exc}_{x}(i) = 0, \forall i \in V \setminus \{s, d\}.$$  

The static flow $x$ is called feasible if

$$0 \leq x(e) \leq b(e), \forall e \in A.$$
The value of $x$ is defined as:

\[(2.4) \quad \text{val}(x) = \text{exc}_x(d).\]

If $\text{exc}_i(i) = 0$ for all $i \in V$, then $x$ is called a circulation.

The famous maximum (static) flow problem aims at finding a feasible static flow $x$ that maximizes $\text{val}(x)$. For details, we refer to Ahuja et al. [1] and Dhamala et al. [5].

The above-discussed formulation is the arc-flow formulation of a static flow. An alternative to this approach is the path and cycle flow formulation. Let $\Gamma$ be the collection of all the $s$-$d$ paths and $C$ be the collection of cycles of the network. Let $f(\gamma)$ and $f(C)$ be the flows in $\gamma \in \Gamma$ and $C \in C$, then the arc flow is

\[(2.5) \quad x(e) = \sum_{\gamma \in \Gamma} \delta_e(\gamma) f(\gamma) + \sum_{C \in C} \delta_e(C) f(C)\]

where $\delta_e(\gamma) = 1$ if $e \in \gamma$, and zero otherwise. Similarly, $\delta_e(C) = 1$ if $e \in C$, and zero otherwise. The relation (2.5) determines $x$ uniquely if path and cycle flows are given. Conversely, given an arc flow $x$, we can find path ($s$-$d$ path) and cycle flow $f$ (not necessarily unique) such that (2.5) is satisfied. This is called the flow decomposition of $x$ into path and cycle flows. For more details, we refer to Ahuja et al. [1].

A very important concept in flow optimizations is the residual network. We denote the static residual network of $N = (V, A, b, \tau, s, d)$ with respect to the flow $x$ by $N(x)$. $N(x)$ has the same vertex set $V$ and an arc set $A(x) = A_F(x) \cup A_B(x)$ where $A_F(x) = \{(i, j) \mid x(i, j) < b(i, j)\}$ and $A_B(x) = \{(j, i) \mid x(i, j) > 0\}$. For $(j, i) \in A_B(x)$, $\tau(j, i) = -\tau(i, j)$. In the residual network $N(x)$, we define the residual capacity $b_x : A(x) \rightarrow \mathbb{R}$ by

\[b_x(i, j) = \begin{cases} b(i, j) - x(i, j) & \text{if } (i, j) \in A_F(x) \\ x(j, i) & \text{if } (i, j) \in A_B(x) \end{cases}\]

The relation of the static flow with the residual network is that whenever there exists a path from the source $s$ to the sink $d$ in the residual network, the value of the flow can be increased.

A dynamic flow $\Phi$ with time horizon $T$ consists of Lebesgue-integrable functions $\Phi_e : [0, T) \rightarrow \mathbb{R}_{\geq 0}$ for each arc $e \in A$ such that $\Phi_e(\theta) = 0$ for $\theta \geq T - \tau(e)$. $\Phi_e(\theta)$ can be realized as the rate of flow entering $e$ at time $\theta$. The flow entering the tail $i$ of the arc $e = (i, j)$ at time $\theta$ reach the head $j$ of $e$ at time $\theta + \tau_e$. For each $i \in V$, we define the excess of node $i$ induced by $\Phi$ at time $\theta$ as:

\[(2.6) \quad \text{exc}_\Phi(i, \theta) = \sum_{e \in A_{i}^{+}} \int_{0}^{\theta - \tau(e)} \Phi_e(\sigma) d\sigma - \sum_{e \in A_{i}^{-}} \int_{0}^{\theta} \Phi_e(\sigma) d\sigma\]

which is the net amount of flow that enters node $i$ up to time $\theta$.

**Definition 2.2.** A feasible dynamic $s$-$d$ flow $(s, d \in V$ and $s \neq d)$ satisfies:

\[(2.7) \quad \text{exc}_\Phi(i, \theta) \geq 0 \quad \forall \theta \in [0, T), \quad \forall i \in V \setminus \{s\}\]

\[(2.8) \quad \text{exc}_\Phi(i, T) = 0, \quad \forall i \in V \setminus \{s, d\}\]
The value of the dynamic flow $\Phi$ at time $\theta$ is
\[ \text{val}_\theta(\Phi) = \text{ex}_\Phi(d, \theta) \]
and the total value of the dynamic flow $\Phi$ is:
\[ \text{val}(\Phi) = \text{val}_T(\Phi) = \text{ex}_\Phi(d, T) \]

For more details, we refer to Skutella [38].

In the course of designing efficient algorithms related to a dynamic flow, a dynamic flow is represented as what is known as temporally repeated flow. Given a feasible static flow $x$ and a time horizon $T$, a flow decomposition on $x$ gives a set of paths $\Gamma$ with flow $f(\gamma)$ for each $\gamma \in \Gamma$. Flow is sent along $\gamma$ at a constant rate $f(\gamma)$ from time 0 to $\max\{T - \tau(\gamma), 0\}$, where $\tau(\gamma) = \sum_{e \in \gamma} \tau(e)$ is the travel time on path $\gamma$. In this way, the dynamic flow is obtained as described in the following equation
\[ (2.10) \quad \Phi_e(\theta) = \sum_{\gamma \in \Gamma_e(\theta)} f(\gamma), \quad \forall e = (i, j) \in A, \, \theta \in [0, T) \]
where $\Gamma_e(\theta) = \{ \gamma \in \Gamma | e \in P \text{ and } \tau(\gamma_s, i) \leq \theta \text{ and } \tau(\gamma_j, d) < T - \theta \}$. Here, $\tau(\gamma_s, i)$ is the sum of times of the arcs from node $s$ to node $i$ on path $\gamma$ and $\tau(\gamma_j, d)$ is the sum of times of the arcs from node $j$ to node $d$ on path $\gamma$.

Given a time horizon $T$, the maximum dynamic problem seeks to find a dynamic flow $\Phi$ which maximizes $\text{val}_T(\Phi)$. Using temporally repeated flows, Ford and Fulkerson [8] showed that finding a maximum dynamic flow is equivalent to finding a minimum circulation $x$ that minimizes $\sum_{e \in A} \tau(e)x(e) - T \cdot \text{val}(x)$ adding an arc $(d, s)$, to the network, with infinite capacity and $-T$ cost(time). From the flow decomposition of $x$, one can find the dynamic maximum flow in the temporally repeated form using equation (2.10).

Given a time horizon $T$, we say that a dynamic flow has the earliest arrival property if as much flow as possible arrives at the sink at each time $\theta < T$ and the corresponding flow is called earliest arrival flow. Every earliest arrival flow is also a maximum dynamic flow but the converse is not true in general (Ruzika et al. [34]).

A problem closely related with the maximum dynamic flow problem is the quickest flow problem which seeks to find dynamic flow with minimum time horizon $T^*$ needed to send a given amount of flow $F$ from the source $s$ to the sink $d$.

Using the idea of finding a dynamic flow with the temporal repetition of the static flow, the following mathematical programming formulation of the problem is useful in designing algorithms to find a quickest flow.

**Theorem 2.3** (Lin and Jaillet [18]). The quickest flow problem can be formulated as the following fractional programming problem:
\begin{align}
(2.11) & \quad \min_{v} \frac{F + \sum_{e \in A} \tau(e) x(e)}{v} \\
(2.12) & \quad \text{s.t. } \sum_{e \in A^{\text{out}}_i} x(e) - \sum_{e \in A^{\text{in}}_i} x(e) = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{otherwise} \end{cases} \\
(2.13) & \quad 0 \leq x(e) \leq b(e), \forall e \in A.
\end{align}

Observing that if \( v \) is fixed in the above formulation, it becomes a min-cost flow problem with supply at \( s \) and demand at \( d \) both equal to \( v \), they realize that the quickest flow problem is a parametric min-cost flow problem with respect to \( v \). Deriving optimality conditions on this basis, they design a cost-scaling algorithm to solve the quickest flow problem with polynomial time complexity \( O(n^3 \log(nC)) \) of a min-cost flow problem, where \( C \) is the maximum arc cost. Stepping on their approach Saho and Sigeno [35] design a cancel-and-tighten algorithm which runs in strongly polynomial time, \( O(nm^2 \log^2 n) \), to solve the quickest flow problem.

3. Combining location decisions with the quickest flow

If a facility is placed on an arc of a network, it reduces the capacity of corresponding arc affecting the decisions related to flow. Hamacher et al. [11] model such problems so that there is the least reduction in the maximum flow value because of the placement of the facility.

**Definition 3.1** (Maximum static and dynamic FlowLoc). Let \( N = (V, A, b, \tau, s, d) \) be a network with set of all feasible locations \( L \subseteq A \), set of all facilities \( P \), the size of the facilities \( r : P \rightarrow \mathbb{N} \) and the number of facilities that can be placed on the possible locations \( \nu : L \rightarrow \mathbb{N} \). The maximum static FlowLoc problem asks for an allocation \( \pi : P \rightarrow L \), such that the static \( s-d \) flow value in the network \( N^\pi = (V, A, b^\pi, s, d) \) is maximized where the capacity function \( b^\pi \) is defined as \( b^\pi(e) = b(e) - \max\{r(p) : p \in P \text{ and } \pi(p) = e\} \). In the above setting, if the dynamic \( s-d \) flow value is to be maximized, it is called a maximum dynamic FlowLoc problem. If \( |P| = 1 \), the corresponding problem is called a single facility maximum static (dynamic) FlowLoc problem and if \( |P| = q > 1 \), then it is referred to as a static (dynamic) multi facility FlowLoc or a \( q \)-FlowLoc problem.

Because of the reduction in the arc capacity, placing a facility on an arc of a network may result into the increase in the time to transfer a given amount of flow from the source to the sink. In the following definition, we consider the problem of locating the facilities so that the increase in the quickest time is minimum.

**Definition 3.2** (Quickest FlowLoc). Given a network \( N = (V, A, b, \tau, s, d) \), a supply \( F \) at \( s \), set of feasible locations \( L \subseteq A \), the set of all facilities \( P \), the size of the facilities \( r : P \rightarrow \mathbb{N} \), the number of facilities that can be placed on the possible locations \( \nu : L \rightarrow \mathbb{N} \), the quickest FlowLoc problem seeks for an allocation \( \pi : P \rightarrow L \) of the facilities to the edges, such that the quickest time to transport \( F \) from \( s \) to \( d \) on the network \( N^\pi = (V, A, b^\pi, \tau, s, d) \), where \( b^\pi(e) = b(e) - \max\{r(p) : p \in P \text{ and } \pi(p) = e\} \), is minimized.

**Remark 1.** In our considerations, the set of feasible locations \( L \) and the set of facilities \( P \) are to be given in such a way that
Figure 1. Evacuation network $N$ with arc labels (capacity, travel time)

(i) $|P| \leq \sum_{e \in L} v(e)$ and
(ii) $r(p) \leq \min \{b(e) : e \in L\}, \ \forall p \in P$

The following example gives the comparison of location decisions on arcs under different flow decisions.

Example 1. Consider the evacuation network depicted in Figure 1. The pair of numbers on each arc represents capacity and travel time related to the arc. Let $P = \{p\}, r(p) = 1, L = \{(2, 3), (2, 4)\}$, i.e. a facility $p$ of size $r(p) = 1$ is to be placed on one of the arcs in $L$. If the facility is placed on $(2, 3)$, the maximum static flow value is 7 (4 along the path $1 - 2 - 4$ and 3 along the path $1 - 3 - 4$), while if it is placed on $(2, 4)$, the maximum static flow value is 6 (3 along the path $1 - 2 - 4$ and $1 - 3 - 4$ each). Hence, $\pi(p) = (2, 3)$ is the maximum static FlowLoc decision, which does not consider the time factor associated with arcs.

Table 1 shows that maximum dynamic FlowLoc decisions depend on the time horizon $T$. In the continuous time setting, when $T = 4$, if no facility is placed, a flow of value 1 can reach the sink using only the path $1 - 2 - 3 - 4$. So if the facility is placed on the arc $(2, 3)$, this path gets obstructed and no flow can reach the sink, so $(2, 4)$ is the optimal location in this case. When $T = 5$, if the facility is placed on $(2, 3)$, the flow of value 4 can reach the sink via path $1 - 2 - 4$, while the flow of value 5 can reach the sink if we place the facility on $(2, 4)$ using the path $1 - 2 - 3 - 4$ with flow value 1 twice and path $1 - 2 - 4$ with flow value 3 once. Table 2 shows that the quickest FlowLoc decisions depend on the flow value $F$ to be transferred from $s$ to $d$. The calculation of the quickest time is done using cost scaling algorithm by Lin and Jaillet [18] (see Section 3.1.1). As expected, the maximum dynamic FlowLoc and quickest FlowLoc decisions are related with each other in some sense.

3.1. Single facility quickest FlowLoc. In this section, we design efficient algorithms to solve a single facility ($|P| = 1$) quickest FlowLoc problem. To set up a background, we discuss the single facility static and dynamic FlowLoc problems.

To solve the single facility static FlowLoc problem, Hamacher et al. [11] give three algorithms. The main idea is to iterate over the arcs in $L$, place the facility, calculate the maximum flow value and finally choose the arc which gives the greatest maximum flow value to place the facility. With preflow push algorithm to perform
maximal flow calculations, the time complexity of such an algorithm, as realized in [11], is $O(|L|^3)$.

Observation 1. Using the algorithm given by Orlin [20] to find the maximal flow, the time complexity of single facility static FlowLoc problem can be reduced to $O(|L|mn)$.

In case of a single facility dynamic FlowLoc, a similar idea can be used calculating a maximal dynamic flow with the help of temporally repeated static flow. Given a time horizon $T$, to calculate the temporally repeated static flow corresponding to the maximal dynamic flow, one can take $\tau$ as cost, add an arc $(d, s)$ in the network with infinite capacity and cost $-T$, and calculate minimum cost circulation in the modified network. As suggested in [11], we present Algorithm 1 to solve the single facility dynamic FlowLoc.

Observation 2. Using the dual network simplex algorithm presented in Armstrong and Jin [3] for solving the minimum cost flow problem, Algorithm 1 solves the single facility dynamic FlowLoc problem in $O(|L|mn(m + n \log n) \log n)$ time. In a series parallel graph (Ruzika et al. [32]), the maximal dynamic flow problem with time horizon $T$ can be solved in $O(mn + m \log m)$ time, using greedy approach, by sending the flow iteratively through the $s-d$ path with the minimal time and removing the saturated arc, considering only the paths with time not exceeding $T$. Thus, in a series parallel graph, the single facility maximal dynamic FlowLoc problem can be solved in $O(|L|(mn + m \log m))$ time. Moreover, in such a graph, a maximal flow has also the earliest arrival property which requires the flow to be maximized at each period of time. Thus, maximal dynamic FlowLoc decisions under earliest arrival flow in a series parallel graph can also be solved with the same time complexity, although finding earliest arrival flow in a general graph has a pseudopolynomial time complexity.
Algorithm 1: Single facility maximum dynamic FlowLoc

Input: Directed network \( N = (V, A, b, \tau, s, d) \), the set of possible locations \( L \), time horizon \( T \), size \( r \) of the facility

Output: Location \( loc \) of the facility, the static flow \( x \) corresponding to the maximum dynamic flow

1. Add an arc \((d, s)\) with infinite capacity and cost \(-T\) to \( N \) and consider \( \tau \) as cost to obtain a network \( N^c \)
2. \( curr\_max\_flow = -1 \)
3. for \( l \in L \) do
   4. \( b(l) = b(l) - r \)
   5. \( x' = \text{min-cost circulation in } N^c \)
   6. \( new\_max\_flow = -\text{cost of } x' \)
   7. if \( new\_max\_flow > curr\_max\_flow \) then
      8. \( curr\_max\_flow = new\_max\_flow \)
      9. \( loc = l \)
   10. \( x = \text{restriction of } x' \) to \( N \)
11. end
12. \( b(l) = b(l) + r \)
13. end
14. return \( loc, x \)

Before proceeding further, we summarize two algorithms to solve the quickest flow problem, which are central in designing algorithms to solve quickest FlowLoc problems.

3.1.1. Cost scaling algorithm (Lin and Jaillet, [18]). Given \( N = (V, A, b, \tau, s, d) \) as an evacuation network with a supply \( F \) at \( s \), let \( x \) be a static flow with value \( v \). Node potentials \( \rho \) are introduced and the reduced cost \( c_e = \rho(j) - \rho(i) + \tau(e) \) is calculated for each arc \( e = (i, j) \in N(x) \), the residual network corresponding to \( x \). When \( c_e > -\epsilon, \forall e \in N(x) \), the obtained flow \( x \) is called \( \epsilon \)-optimal. The algorithm is briefly described in the following steps.

1. Initialize: \( \rho(u) = 0, \forall u \in V, x(e) = 0, \forall e \in A \), and \( \epsilon = C = \max_{e \in A} \{\tau(e)\} \).
2. Refine: The \( 2\epsilon \)-optimal flow is modified to an \( \epsilon \)-optimal one by assigning the flow in the arcs of \( N \) with \( c_e < 0 \) to their capacity, assigning zero flow in the arcs with \( c_e > 0 \), then pushing flows from nodes with excess flow through the arcs in the residual network \( N(x) \), relabeling their potential if required.
3. Reduce Gap: Set extra flow at \( s \) and push the admissible flow ultimately to \( d \) with arcs in \( N(x) \), and relabel the potential of nodes if required to reduce the gap between \( T = [F + \sum_{e \in A} \tau(e)x(e)]/v \) and \( \rho(s) - \rho(d) \) by at least \( 7n\epsilon \).

After Step 3, \( \epsilon \) is scaled by \( 1/2 \), and Steps 2 and 3 are repeated unless it becomes less than \( 1/8n \).

4. Saturate: If \( T \), obtained from the above-mentioned scaling phases, is more than the time (cost) in a shortest simple path from \( s \) to \( d \) in the residual network \( N(x) \), the flow is saturated by sending maximum flow from \( s \) to \( d \)
in a subnetwork $N'$, formed by only those arcs which are on some shortest path from $s$ to $d$ in $N(x)$.

The time complexity of the cost-scaling algorithm is $O(n^3 \log(nC))$. For details, we refer to [18].

3.1.2. Cancel-and-tighten algorithm (Saho and Sigeno, [35]). The main idea of the algorithm is to modify the cost scaling algorithm replacing Step 2 with Cancel and Tighten steps.

- **Cancel:** Find a cycle in $N(x)$ with only admissible arcs (an arc $e \in N(x)$ is admissible if its reduced cost $c_e < 0$) and push a flow equal to minimum residual capacity of its arcs. Repeat the process until there remains no such cycle.
- **Tighten:** For each node $u$, compute the maximum length $h(u)$ from nodes with no entering admissible arc. Replace $\rho(u)$ by $\rho(u) + \frac{\epsilon}{n}h(u)$ and reduce $\epsilon$ to $(1 - \frac{1}{n})\epsilon$.

Cancel Step and Tighten Steps are repeated iteratively until $\epsilon$ reduces to $\epsilon/2$. Then Reduce Gap step reduces the gap between $T = \frac{F + \sum_{e \in A} \tau(e)x(e)}{v}$ and $\rho(s) - \rho(d)$ by at least $(3n + 1)\epsilon$ in this case.

The above-mentioned steps are performed until $\epsilon$ becomes smaller than $1/4n$, and finally Saturate step is performed as in cost scaling algorithm. The complexity of this algorithm is $O(nm^2 \log^2 n)$.

Now, we construct two algorithms for the single facility quickest FlowLoc problem. Algorithm 2 iterates through all possible locations $l \in L$, determines the quickest time if location $l$ hosts the facility and finds the optimal location for the single facility by comparing all those quickest times. It also records the quickest flow, and the quickest time after placing the facility.

**Algorithm 2:** Single facility quickest FlowLoc I

| Input | Directed network $N = (V, A, b, \tau, s, d)$, the set of possible locations $L$, supply $F$ at $s$, size $r$ of the facility |
|-------|----------------------------------------------------------------------|
| Output| Location $loc$ of the facility, the corresponding static flow $x$, the corresponding quickest time $T$ |

1. $T = \infty$
2. **for** $l \in L$ **do**
3. $b(l) = b(l) - r$
4. $new$-quickest-time = the quickest time in the modified network
5. **if** $new$-quickest-time < $T$ **then**
6. $T = new$-quickest-time
7. $loc = l$
8. $x =$ static flow corresponding to the quickest flow
9. **end**
10. $b(l) = b(l) + r$
11. **end**
12. **return** $loc, x, T$

Algorithm 2 performs the quickest flow computations $|L|$ times. If we perform a single quickest flow computation before going through Algorithm 2, and find that...
an arc in $L$ has residual capacity enough to accommodate the given facility, we can get rid of $|L| - 1$ quickest flow computations in Algorithm 2. Algorithm 3 addresses this issue.

**Algorithm 3:** Single facility quickest FlowLoc II

**Input:** Directed network $N = (V, A, b, c, s, d)$, the set of possible locations $L$, supply $F$ at $s$, size $r$ of the facility

**Output:** Location $loc$ of the facility, the corresponding static flow $x$, and the quickest time $T$

1. $x =$ static flow corresponding to the quickest flow in the network $N$
2. $T =$ the corresponding quickest time
3. $e = \arg \max \{b(l) - x(l) | l \in L\}$
4. If $b(e) - x(e) \geq r$ then
   5. $loc = e$
   6. $b(e) = b(e) - r$
5. Else
   6. Algorithm 2
7. End
8. Return location $loc, x, T$

**Theorem 3.3.** The single facility quickest FlowLoc problem can be solved in strongly polynomial time.

**Proof.** In the worst case, Algorithm 2 or Algorithm 3 iterates the quickest flow computations $|L|$ times. The cancel-and-tighten algorithm by Saho and Shigeno [35](see Section 3.1.2), the quickest flow problem can be solved in $O(nm^2 \log^2 n)$ time. Hence, the single facility quickest FlowLoc problem can be solved in $O(|L|nm^2 \log^2 n)$ time.

**Example 2.** To illustrate the working of Algorithm 2 and Algorithm 3, we consider the network depicted in Figure 1 with $L = \{(1, 2), (1, 3), (2, 3)\}, r = 1, F = 11$. In Algorithm 2, we take $curr_{quickest\_time} = \infty$. In the first iteration, we take $l = (1, 2)$, reduce its capacity 4 by 1 and calculate the quickest time which is 6.33 < $\infty$. So $loc = (1, 2)$ and the capacity of (1, 2) is retained to 4. In this way, after the third iteration, we get $loc = (1, 3)$ after three quickest flow calculations. However, in Algorithm 3, we calculate the static flow $x$ associated with the quickest flow in the beginning (see the first table in Example 3) and see that $b(1, 3) - x(1, 3) = 1 \geq r$ so that $loc = (1, 3)$.

**Observation 3.** Let $|P > 1$ and $\nu(e) \geq |P|, \forall e \in L, p^* = \arg \max \{r(p) : p \in P\}$. If $e^* \in L$ is the single facility location, taken $p^*$ as the single facility, then $\pi(p) = e^*, \forall p \in P$.

3.2. **Multi-facility quickest FlowLoc.** Now we consider the quickest FlowLoc problem for $|P| > 1$. The idea of the single facility case (i.e. iterating over all the possibilities to locate facilities) can be carried over to the multiple facility case also. As described in Hamacher et. al. [11], this does not lead to a polynomial
algorithm even in case of static FlowLoc, the exception being the case mentioned in Observation 3. The following result is crucial in this regard.

**Theorem 3.4** (Hamacher et al.,[11]). *There is no polynomial time $\alpha$-approximation algorithm for the multi-facility maximum static FlowLoc problem with a finite constant $\alpha$ unless $P = NP$.*

As we have seen that the multi-facility static maximum FlowLoc problem is $NP$-hard, we realize the hardness of the multi-facility quickest FlowLoc problem in the following lemma.

**Lemma 3.5.** For $F > 0$, if $\tau_e = 0 \ \forall e \in A$, the quickest flow problem is equivalent to the maximum static flow problem.

*Proof.* The maximum static flow problem can be stated as:

\[
\max \quad v
\]

\[
\sum_{e \in A^{out}} x_e - \sum_{e \in A^{in}} x_e = \begin{cases} 
v & \text{if } w = s \\
-v & \text{if } w = d \\
0 & \text{if } w \notin \{s,d\}
\end{cases}
\]

\[
0 \leq x_e \leq b_e \ \forall e \in A
\]

The objective function (3.1) replaced by $F + \sum_{e \in A} x_e \tau_e$ with the same constraints gives the quickest flow problem because of Theorem 2.3. If $\tau_e = 0 \ \forall e \in A$, the quickest flow problem reduces to minimize $F/v$ subject to the constraints (3.2) and (3.3). But for a fixed $F > 0$, minimizers of $F/v$ will maximize $v$. □

**Theorem 3.6.** *There is no polynomial time $\alpha$-approximation algorithm to solve the multifacility quickest FlowLoc problem unless $P = NP$.*

*Proof.* Suppose that there is a polynomial time $\alpha$-approximation algorithm to solve the multifacility quickest FlowLoc problem for $\alpha < \infty$. According to Lemma 3.5, the maximum static flow problem is a special case of the quickest flow problem. This implies that there exists such an algorithm for multi-facility static FlowLoc problem which contradicts Theorem 3.4. □

Because of Theorem 3.6, we design a polynomial time heuristic to solve the multi-facility quickest FlowLoc problem presenting its mixed integer programming formulation. The mathematical programming formulation of the multi-facility quickest FlowLoc problem, based on Theorem 2.3, is as follows.
The variables and constants used in the model are described as follows.

**Variables**

\( x_e \) = static flow corresponding to the quickest flow in \( e \in A \)

\( y_{ep} = \begin{cases} 1 & \text{if the facility } p \text{ is placed on } e \in L \\ 0 & \text{if the facility } p \text{ is not placed on } e \in L \end{cases} \)

**Constants**

\( r_p = r(p) \), the size of the facility \( p \)

\( b_e = b(e) \), the capacity of \( e \in A \)

\( \nu_e = \nu(e) \) = the number of facilities that can be placed on \( e \in L \)

Constraints (3.5) and (3.7) are conditions for a static flow. Constraints (3.6) reduce the capacity of \( e \) by \( r_p \) if the facility \( p \) is placed on \( e \). Constraints (3.8) state that each facility has to be placed in exactly one arc, and constraints (3.9) bound the number of facilities on an arc by the admissible number of facilities on it.

The objective function of the above problem is not linear. To make it linear, we put \( 1/v = \theta \), and \( x_e \theta = \xi_e \). As a result the problem becomes

\[
\begin{align*}
\text{(3.11)} & \quad \min \quad F + \frac{\sum_{e \in A} \tau_e x_e}{v} \\
\text{(3.12)} & \quad \sum_{e \in A^{\text{out}}_i} \xi_e - \sum_{e \in A^{\text{in}}_i} \xi_e = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{if } i \in V \setminus \{s, d\} \end{cases} \\
\text{(3.13)} & \quad \xi_e + \theta y_{ep} r_p \leq b_e, \forall e \in L, p \in P \\
\text{(3.14)} & \quad 0 \leq \xi_e \leq b_e, \forall e \in A \\
\text{(3.15)} & \quad \sum_{e \in L} y_{ep} = 1, \forall p \in P \\
\text{(3.16)} & \quad \sum_{p \in P} y_{ep} \leq \nu_e, \forall e \in L \\
\text{(3.17)} & \quad y_{ep} \in \{0, 1\}, \forall e \in L, p \in P
\end{align*}
\]

However, the set of constraints (3.13) are not linear. If one wants to use a linear mixed integer programming solver to solve the model, one can linearize them using
the idea given in Torres [39], replacing (3.13) with the following constraints \( \forall e \in L, p \in P \)

\[
\begin{align*}
(3.18) & \quad \xi_e + \zeta_{ep} r_p \leq b_e \theta \\
(3.19) & \quad \zeta_{ep} \leq My_{ep} \\
(3.20) & \quad \zeta_{ep} \leq \theta \\
(3.21) & \quad \zeta_{ep} \geq \theta - (1-y_{ep})M \\
(3.22) & \quad \zeta_{ep} \geq 0
\end{align*}
\]

where \( M \) is an upper bound on the value of \( \theta \) which can be taken 1 if there is at least one path from the source to sink with positive integral capacities and \( F \) is also a positive integer, because \( v \) is at least 1 in such cases.

Now, we construct a polynomial time heuristic, in Algorithm 4, to solve the problem that works well in practice. First of all, the facilities are sorted in decreasing order of their sizes. Then the quickest flow calculation is done (polynomial time algorithms for such calculations exist, see Section 3.1.1 and 3.1.2) and the residual capacities of the arcs in \( L \) are calculated. Then, first \( \nu(e) \) facilities are placed on the arc \( e \) with the largest residual capacity, and \( e \) is removed from \( L \). The process is repeated until all the facilities are allocated to some or all arcs in \( L \). If the residual capacity of \( e \) is less than the size of the largest facility hosted by it, the quickest flow is recalculated in Line 15.

**Algorithm 4:** Multi facility quickest FlowLoc heuristic

**Input:** Directed network \( N = (V, A, b, \tau, s, d) \), the set of possible locations \( L \), supply \( F \) at the source \( s \), set of facilities \( P \) with size \( r : P \to \mathbb{N} \)

**Output:** Allocation \( \pi : P \to L \), the quickest time \( T \) after allocation

1. sort facilities according to their size \( r(p_1) \geq r(p_2) \geq \cdots \geq r(p_q) \)
2. \( x = \) static flow corresponding to the quickest flow
3. \( T = \) the corresponding quickest time
4. \( i = 1 \)
5. **while** \( i \leq q \) **do**
6. \( e = \arg \max \{ b(l) - x(l) \} | l \in L \} \)
7. **for** \( j = 1 \ to \ j = \nu(e) \) **do**
8. **if** \( i + j - 1 \leq q \) **then**
9. \( \pi(p_{i+j-1}) = e \)
10. **end**
11. **end**
12. \( L = L \setminus \{e\} \)
13. \( b(e) = b(e) - r(p_i) \)
14. **if** \( b(e) - x(e) + r(p_i) < r(p_i) \) **then**
15. \( x = \) static flow corresponding to the quickest flow with modified \( b \)
16. \( T = \) the corresponding quickest time
17. **end**
18. \( i = i + \nu(e) \)
19. **end**
20. **return** \( \pi, x, T \)
Example 3. To illustrate Algorithm 4, we consider the network given in Figure 1. Let $L = \{(2,1), (2,4), (1,3), (3,4)\}$ with $\nu(2,1) = 1, \nu(2,4) = 2, \nu(1,3) = 1, \nu(3,4) = 3$ and $P = \{f_1, f_2, f_3, f_4\}$ with $r(f_1) = 1, r(f_2) = 3, r(f_3) = 2, r(f_4) = 1$.

First of all, we order the facilities in the decreasing order of their size, i.e. $p_1 = f_2, p_2 = f_3, p_3 = f_1, p_4 = f_4$ so that $r(p_1) = 3, r(p_2) = 2, r(p_3) = 1, r(p_4) = 1$.

The static flow corresponding to the quickest flow is given in the following table.

| arc   | (1,2) | (1,3) | (2,1) | (2,4) | (2,3) | (3,1) | (3,2) | (3,4) | (4,2) | (4,3) |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| $b$   | 4     | 3     | 3     | 4     | 1     | 3     | 1     | 3     | 3     | 1     |
| $x$   | 4     | 2     | 0     | 3     | 1     | 0     | 0     | 3     | 0     | 0     |
| $b(l) - x(l) | l \in L | 1     | 3     | 1     | 1     | 0     | 0     | 3     | 0     | 0     |

$T = 6$
$q = 4$
$i = 1 < q$

First iteration:

$e = \max \{b(e) - x(e) | e \in L\} = (2,1)$

$j = 1$
$i + j - 1 = 1 + 1 - 1 = 1$

$\pi(p_1) = (2,1)$
$L = \{(2,4), (1,3), (3,4)\}$

$b(e) = 3 - 3 = 0$

$b(e) - x(e) + r(p_i) = 0 - 0 + 3 \not< r(p_1)$

$i = 1 + 1 = 2 < q$

Second iteration:

$e = \max \{b(e) - x(e) | e \in L\} = (2,4)$ (We may take $e = (1,3)$ also.)

$j = 1, 2$

$i + j - 1 = 2 + 1 - 1, 2 + 2 - 1 = 2, 3$

$\pi(p_2) = (2,4)$
$\pi(p_3) = (2,4)$
$L = \{(1,3), (3,4)\}$

$b(e) = 4 - 2 = 2$

$b(e) - x(e) + r(p_i) = 2 - 3 + 2 = 1 < r(p_i) = 2$

We recalculate $x$

| arc   | (1,2) | (1,3) | (2,1) | (2,4) | (2,3) | (3,1) | (3,2) | (3,4) | (4,2) | (4,3) |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| $b$   | 4     | 3     | 0     | 2     | 1     | 3     | 1     | 3     | 3     | 1     |
| $x$   | 3     | 2     | 0     | 2     | 1     | 0     | 0     | 3     | 0     | 0     |
| $b(l) - x(l) | l \in L | 1     | 1     | 0     | 0     | 3     | 0     | 0     | 0     | 0     |

$T = 6.4$
$i = 2 + 2 = 4 = q$

Third iteration:

$e = \max \{b(e) - x(e) | e \in L\} = (1,3)$

$j = 1$

$i + j - 1 = 4 + 1 - 1 = 4$

$\pi(p_4) = (1,3)$
Figure 2. A randomly generated weakly connected directed graph with $n = 50$ and edge density 20%

$L = \{(3, 4)\}$

$b(e) = 3 - 1 = 2$

$b(e) - x(e) + r(p_i) = 2 - 2 + 1 = 1 \neq r(p_i) = r(p_4) = 1$

$i = 4 + 1 = 5 > q$

Since $i > q$, the algorithm terminates and the solution is: $\pi(f_1) = \pi(p_3) = (2, 4)$, $\pi(f_2) = \pi(p_1) = (2, 1)$, $\pi(f_3) = \pi(p_2) = (2, 4)$, $\pi(f_4) = \pi(p_4) = (1, 3)$. The static flow corresponding to the quickest flow $x$ is given in the following table with the quickest time $T = 6.4$.

| arc  | (1, 2) | (1, 3) | (2, 1) | (2, 4) | (2, 3) | (3, 1) | (3, 2) | (3, 4) | (4, 2) | (4, 3) |
|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| $b$  | 4      | 2      | 0      | 2      | 1      | 3      | 1      | 3      | 3      | 1      |
| $x$  | 3      | 2      | 0      | 2      | 1      | 0      | 0      | 3      | 0      | 0      |

The quickest time $T$ calculated by MILP solver for this example is 6.2 which is very close to result obtained by the heuristic. Worth noting, in this example, is that if $e = (1, 3)$ is chosen in the second iteration, the result of the heuristic coincides with that of the MILP solver.

3.3. Computational experiment. For the computations, we have generated random weakly connected directed graphs with $n$ nodes and 20% edge density. One such graph with $n = 50$ is shown in Figure 2. The capacity of each arc is taken randomly between 1 to 5 units of flow (e.g. cars) per second, and the travel time on each arc between 1 minute to 10 minutes. A random sample of 20% of the arcs is chosen as $L$, and $1 \leq \nu(l) \leq 5, \forall l \in L$. Taking $F = 10,000$, with $n \in \{50, 100, 200, 400\}$
and \(|P| \in \{50, 75, 100\}\), we have taken the average running time on 10 graphs per instance.

The running times of Algorithm 4 with the mixed integer programming formulation are recorded in Table 4. For an instance of \(n = 400\), \(|P| = 50\), the MILP solver took more than 45 minutes of running time and we have not recorded the time in the table. For an instance of \(n = 1000\), \(|P| = 50\), Algorithm 4 took 37 seconds and CBC solver was unable to find the solution.

**Table 4. Running time (in seconds) of Algorithm 4 and mixed integer programming formulation of multi-facility quickest FlowLoc problem**

| \(n\) | \(|P| = 50\) | \(|P| = 75\) | \(|P| = 100\) |
|-------|-------------|-------------|-------------|
| 50    | Algorithm 4 | 0.32        | 0.23        | 0.22        |
|       | MILP        | 4.46        | 9.05        | 14.27       |
| 100   | Algorithm 4 | 0.49        | 0.48        | 0.45        |
|       | MILP        | 27.76       | 50.71       | 77.88       |
| 200   | Algorithm 4 | 1.48        | 1.45        | 1.46        |
|       | MILP        | 232.47      | 463.96      | 714.74      |
| 400   | Algorithm 4 | 5.44        | 5.46        | 5.40        |
|       | MILP        | -           | -           | -           |

In each instance, the percentage deviation of the objective function value from the MILP objective function value is found to be zero, i.e. all the objective function values match in the random experiments. It means that the proposed algorithm works well in practice.

For the implementation of algorithms and solution of mixed integer programming, we have used Python 3.7 on a computer with Intel® Core™ i5, 2.30 GHz processor, 4GB RAM, and 64-bit operating system. The mixed integer program has been solved using CBC (Coin-OR branch and cut) solver.

4. Quickest FlowLoc problem with contraflow

Assuming that the direction of arcs in a directed network can be reversed (i.e. the direction of the traffic flow on a road segment can be reversed), a contraflow problem seeks to choose the ideal direction of arcs to optimize network flows. To solve a dynamic contraflow problem on a network \(N = (V, A, b, \tau, s, d)\), an undirected network \(\bar{N} = (V, \bar{A}, \bar{b}, \bar{\tau}, s, d)\), known as auxiliary network of \(N\), is constructed such that

\[ \bar{A} = \{(u, w) : (u, w) \in A \text{ or } (w, u) \in A\} \]

For each \((u, w) \in \bar{A}\),

\[
\bar{b}(u, w) = b(u, w) + b(w, u) \\
\bar{\tau}(u, w) = \begin{cases} 
\tau(u, w) & \text{if } (u, w) \in A \\
\tau(w, u) & \text{otherwise}
\end{cases}
\]

in which we consider \(b(i, j) = 0\) whenever \((i, j) \notin A\), and vice versa.

Implementation of algorithms to solve various dynamic flow problems on auxiliary network helps to solve the corresponding contraflow problems [32, 24]. For example, to solve the maximum contraflow problem which seeks to maximize the
flow allowing arc reversals in a given network, we solve the maximum flow problem in its auxiliary network. Then, the flow is decomposed into paths and cycles and cycle flows are removed. The analogous procedure to solve the quickest contraflow problem can be found in Pyakurel et al. [28]. The algorithms to solve the contraflow problems not only give optimal flow decisions but also an output which arcs to reverse and which arcs not to reverse. In what follows, the flow in the auxiliary network $\bar{N}$ of $N$ without cycle flows will be referred to as contraflow in $N$.

If we allow reversal of the direction of the usual traffic flow, especially in case of emergency evacuation planning, there may be a significant reduction in the quickest time. The change in the capacity of arcs, in such cases, have also effects on location decisions.

**Definition 4.1 (Maximum static (dynamic) ContraFlowLoc).** Let an evacuation network $N = (V, A, b, \tau, s, d)$ be a network with the set of all feasible locations $L \subseteq A$, set of all facilities $P$, the size of the facilities $r : P \to \mathbb{N}$ and the number of facilities that can be placed on the possible locations $\nu : L \to \mathbb{N}$. The maximum static (dynamic) ContraFlowLoc problem asks for an allocation $\pi : P \to L$, such that the static (dynamic) maximum flow value is maximized after the facility-allocation, allowing arc reversals.

**Definition 4.2 (Quickest ContraFlowLoc).** Given an evacuation network $N = (V, A, b, \tau, s, d)$, supply $F$ at $s$, set of feasible locations $L \subseteq A$, the set of all facilities $P$, the size of the facilities $r : P \to \mathbb{N}$, the number of facilities that can be placed on the possible locations $\nu : L \to \mathbb{N}$, the quickest ContraFlowLoc problem seeks for an allocation $\pi : P \to L$ of the facilities to the edges, such that the quickest time to transport $F$ from $s$ to $d$ is minimized, allowing arc reversals.

**Example 4.** Consider the network given in Figure 1. The labels on arcs denote capacity and travel time respectively. Let the set of feasible locations $L = \{(2,1), (1,3)\}$ and the size of a single facility $r = 2$. If the facility is placed on (2,1), the values of the static maximum flow before and after contraflow configuration are 7 (4 along 1 − 2 − 4 and 3 along 1 − 3 − 4) and 11 (5 along 1 − 2 − 4, 4 along 1 − 3 − 4, 2 along 1 − 3 − 2 − 4) respectively. If the facility is placed on (1,3), the corresponding values are 5 (4 along 1 − 2 − 4, 1 along 1 − 3 − 4) and 11 (7 along...
1−2−4, 4 along 1-3-4) respectively. Thus the static FlowLoc decision before contraflow configuration is (2, 1) and after contraflow configuration is (2, 1) or (1, 3). The decisions with the quickest time before and after contraflow configuration with $F=109$ are listed in Table 5.

| Facility placed on | Quickest time, $F = 109$ |
|-------------------|---------------------------|
|                   | Before contraflow | After contraflow |
| (2, 1)            | 20               | 15               |
| (1, 3)            | 25.8             | 14.27            |

Table 5. Quickest time calculations (cf. Example 4)

To solve the single-facility maximum static(dynamic) FlowLoc problem, we can iteratively choose an arc from $L$, place the facility, reduce its capacity by the size of the facility, calculate the maximum static (dynamic) contraflow value, and choose the arc in which the difference of the maximum contraflow value after placing the facility and without placing the facility on any arc is the least (see also [6]). Here, we present Algorithm 5 to solve the single facility quickest ContraFlowLoc problem, which iteratively chooses an arc from $L$, reduces its capacity by the size of the facility $r$, finds the quickest contraflow and retains its capacity before choosing the next arc. The arc which gives the minimum quickest time after placing the facility on it is chosen as the optimal location.

**Algorithm 5: Single facility quickest ContraFlowLoc I**

**Input**: Directed network $N = (V, A, b, \tau, s, d)$, the set of possible locations $L$, supply at $s = F$, size $r$ of the facility  

**Output**: Location $loc$ of the facility, static contraflow $x$ corresponding to the quickest contraflow, corresponding quickest time $T$, set of arcs to be reversed $R$

1 $T = \infty$
2 for $l \in L$ do 
3 $b(l) = b(l) - r$
4 if $new_{\text{quickest\_time}} < T$ then 
5 $T = new_{\text{quickest\_time}}$
6 $loc = l$
7 $x = \text{the corresponding static contraflow}$
8 end 
9 $b(l) = b(l) + r$
10 end 
11 $R = \{(i,j) \in A | x(i,j) > b(i,j) \text{ if } (i,j) \in A \text{ or } x(i,j) > 0 \text{ if } (i,j) \notin A \}$ 
12 return $loc, x, T, R$

We can improve the practical running time of Algorithm 5 by adapting Algorithm 3 to the contraflow case. After contraflow calculation, if arc $(u,v) \in L$ and its
opposite arc \((v, u) \in A\) together have capacity enough to host the facility, then \((u, v)\) is chosen to locate the facility and we can get rid of the remaining \(|L| - 1\) quickest contraflow calculations of Algorithm 5. The procedure is elucidated in Algorithm 6.

**Algorithm 6: Single facility quickest ContraFlowLoc II**

**Input:** Directed network \(N = (V, A, b, \tau, s, d)\), the set of possible locations \(L\), supply at \(s = F\), size \(r\) of the facility

**Output:** Location \(loc\) of the facility, static contraflow \(x\) corresponding to the quickest contraflow, corresponding quickest time \(T\), set of arcs to be reversed \(R\)

1. \(x = \) static contraflow corresponding to the quickest contraflow \(N\)
2. \(T = \) the corresponding quickest time
3. \((u^*, v^*) = \) arg max\(\{b(u, v) + b(v, u) - x(u, v) - x(v, u) | (u, v) \in L\}\)
4. if \(b(u^*, v^*) + b(v^*, u^*) - x(u^*, v^*) - x(v^*, u^*) \geq r\) then
   5. \(loc = (u^*, v^*)\)
   6. \(b(u^*, v^*) = b(u^*, v^*) - r\)
7. else
8. \ Algorithm 5
9. end
10. \(R = \{(v, u) \in A | x(u, v) > b(u, v) \text{ if } (u, v) \in A \text{ or } x(u, v) > 0 \text{ if } (u, v) \notin A\}\)
11. return \(loc, x, T, R\)

For finding the static contraflow corresponding to the quickest contraflow, we solve the quickest flow problem in the auxiliary network and remove cycle flows (if any) (Pyakurel et al. [28]). Because the size of the facility does not exceed the capacity of an arc in \(L\) (Remark 1), and placing the facility on an arc \((u, v) \in L\) reduces the capacity of \((u, v)\) and \((v, u)\) both in the auxiliary network, and removal of the cycle flows in contraflow calculation, Algorithm 5 and Algorithm 6 find the location \(loc\) with the minimum quickest time. Moreover, the removal cycle flows in a contraflow computation leads either \(x(u, v)\) or \(x(v, u)\) to vanish so that the set of arcs to be reversed \(R\) is well defined. The discussion leads to:

**Lemma 4.3.** Algorithm 5 or Algorithm 6 solves the single facility quickest ContraFlowLoc problem optimally.

**Theorem 4.4.** The single facility quickest ContraFlowLoc problem can be solved in strongly polynomial time.

**Proof.** The complexity of the for loop in Algorithm 5 is dominated by the complexity of the quickest flow calculation which can be done in strongly polynomial time \(O(nm^2 \log^2 n)\). Since the auxiliary network can be formed in linear time, flow decomposition can be done in \(O(nm)\) time (Ahuja et al. [1]), the overall complexity of Algorithm 5 is \(O(|L|nm^2 \log^2 n)\).

Since the multi-facility FlowLoc problems are \(NP\)-hard, the corresponding ContraFlowLoc problems are also \(NP\)-hard. Replacing quickest flow calculations by maximum static(dynamic) contraflow calculations and adjusting capacities accordingly, Algorithm 4, can also be adapted to construct a polynomial time heuristic...
to solve the corresponding multi-facility case. To solve the multi-facility quickest ContraFlowLoc problem, we present such an adaptation in Algorithm 7.

Algorithm 7: Multi facility quickest ContraFlowLoc heuristic

**Input**: Directed network $N = (V, A, b, \tau, s, d)$, the set of possible locations $L$, supply $F$ at the source $s$, set of facilities $P$ with size $r : P \to \mathbb{N}$

**Output**: Allocation $\pi : P \to L$, the quickest time $T$ after allocation, set of arcs to be reversed $R$

1. sort facilities according to their size $r(p_1) \geq r(p_2) \geq \cdots \geq r(p_q)$
2. $x = \text{static contra flow corresponding to the quickest contra flow}$
3. $d(u, v) = b(u, v) + b(v, u) - x(u, v) - x(v, u) \forall (u, v) \in A$
4. $T = \text{the corresponding quickest time}$
5. $i = 1$
6. while $i \leq q$ do
7.  $(u^*, v^*) = \max \{d(u, v)|(u, v) \in L\}$
8.  for $j = 1$ to $j = \nu(u^*, v^*)$ do
9.    if $i + j - 1 \leq q$ then
10.       $\pi(p_{i+j-1}) = (u^*, v^*)$
11.     end
12.  end
13.  $L = L \setminus \{(u^*, v^*)\}$
14.  $b(u^*, v^*) = b(u^*, v^*) - r(p_i)$
15.  if $d(u^*, v^*) + r(p_i) < r(p_i)$ then
16.     $x = \text{static contra flow corresponding to the quickest contra flow with modified } b$
17.  $T = \text{the corresponding quickest time}$
18. end
19. $i = i + \nu(u^*, v^*)$
20. end
21. $R = \{(v, u) \in A| x(u, v) > b(u, v) \text{ if } (u, v) \in A \text{ or } x(u, v) > 0 \text{ if } (u, v) \notin A\}$
22. return $\pi, loc, T, R$

**Example 5.** To illustrate Algorithm 7, we reconsider the problem illustrated in Example 3 with the possibility of arc reversals. The static contra flow corresponding to the quickest contra flow are tabulated in the following table.

| arc  | (1, 2) | (1, 3) | (2, 1) | (2, 4) | (2, 3) | (3, 1) | (3, 2) | (3, 4) | (4, 2) | (4, 3) |
|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| $b$  | 4      | 3      | 3      | 4      | 1      | 3      | 1      | 3      | 3      | 1      |
| $x$  | 7      | 2      | 0      | 5      | 2      | 0      | 0      | 4      | 0      | 0      |
| $d(u, v)|(u, v) \in L$ | 1      | 3      | 1      | 1      | 0      | 0      | 0      | 0      | 0      | 0      |

$T = 5.22$
$q = 4$
$i = 1 < q$

First iteration:
$(u^*, v^*) = \max \{d(u, v)|(u, v) \in L\} = (1, 3)$
$j = 1$
$i + j - 1 = 1 + 1 - 1 = 1$
$\pi(p_1) = (1, 3)$
$L = \{(2, 1), (2, 4), (3, 4)\}$
$b(u^*, v^*) = 3 - 3 = 0$
$d(u^*, v^*) + r(p_i) = 0 + 3 - 2 - 0 + 3 = 4 \n r(p_1) = 3$

Second iteration: $(u^*, v^*) = \max \{d(u,v) | (u,v) \in L\} = (2, 4)$
$j = 1, 2$
$i + j - 1 = 2 + 1 - 1, 2 + 1 - 1 = 2, 3$
$\pi(p_2) = (2, 4)$
$\pi(p_3) = (2, 4)$
$L = \{(2, 1), (3, 4)\}$
$b(u^*, v^*) = 4 - 2 = 2$
$d(u^*, v^*) + r(p_i) = 2 + 3 - 5 - 0 + 2 = 2 \n r(p_2) = 2$

Third iteration:
$(u^*, v^*) = \max \{d(u,v) | (u,v) \in L\} = (2, 1)$
$j = 1$
$i + j - 1 = 4 + 1 - 1 = 4$
$\pi(p_4) = (2, 1)$
$L = \{(3, 4)\}$
$b(u^*, v^*) = 3 - 1 = 2$
$d(u^*, v^*) + r(p_i) = 2 + 4 - 7 - 0 + 1 = 0 \n r(p_4) = 1$
Recalculation of $x$:

| arc | (1, 2) | (1, 3) | (2, 1) | (2, 4) | (2, 3) | (3, 1) | (3, 2) | (3, 4) | (4, 2) | (4, 3) |
|-----|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| $b$  | 4      | 0      | 2      | 2      | 1      | 3      | 1      | 3      | 3      | 1      |
| $x$  | 6      | 2      | 0      | 4      | 2      | 0      | 0      | 4      | 0      | 0      |

$i = 4 + 1 = 5 > q$

The solution is: $\pi(f_1) = \pi(p_3) = (2, 4), \pi(f_2) = \pi(p_1) = (1, 3), \pi(f_3) = \pi(p_2) = (2, 4), \pi(f_4) = \pi(p_4) = (2, 1)$. The static contraflow $x$ corresponding to the quickest contraflow after this allocation is as given in the third iteration. The set of arcs reversed before facility allocation is $\{(2, 1), (3, 2), (4, 2), (4, 3)\}$ while after allocation it becomes $\{(2, 1), (3, 2), (4, 2), (4, 3), (1, 3)\}$. The quickest time is 5.375 which is less than the quickest time 6.2 of the same problem without arc reversals. The significance of the contraflow approach is that the difference between the quickest times before and after arc reversals increase with the growing value of $F$. Some observations of this problem, after facility-allocation are listed in the following table.

| $F$      |  | Quickest time |
|----------|---|---------------|
|          | Before contraflow | After contraflow |
| 100      | 24.2 | 15.33         |
| 1000     | 204.2 | 115.33        |
| 10000    | 2004.2 | 1115.33       |
5. Conclusion

In an effort to combine location decisions with the network flow models, in this paper models to combine quickest flow with location analysis are introduced. With a view to be applied in traffic flow management in emergency evacuation planning, the facility-arc assignments are done so as to affect the quickest time of evacuation the least. Exact polynomial algorithms to assign single facility and polynomial time heuristic algorithms to place multiple facilities on multiple arcs (with and without the possibility of arc reversal) are designed. As in the absence of facility allocations, a significant improvement in the quickest time has been achieved with arc reversals, i.e. with contraflow configuration in the FlowLoc case also. Presented algorithms perform very well in randomly generated graphs, so that they can be used to make facility location decisions in emergency evacuation problems. The algorithms are particularly important when a known volume of evacuees from a danger zone has to be transferred to a safe zone with the least possible interruption in the evacuation time because of facility allocation in some road segments of the transportation network. To the best of our knowledge, the problems and the corresponding algorithms to solve the quickest FlowLoc problems are considered for the first time in this paper.

However, in the problems considered, the locations decisions are made on the basis of quickest flow with a background of maximum flow in a single-source-single-sink network with constant capacity and transit time. Also the number of available facilities does not exceed the number of available locations, and the size of each facility fits in any of the available locations. Hence, the similar problems with other aspects of network flow in more generalized settings can be the natural extensions of the problem.

References

1. R.K. Ahuja, T.L. Magnati, and J.B. Orlin (1988). Network flows, Massachusetts Institute of Technology, Operations Research Center.
2. S. An, N. Cui, X. Li, and Y. Ouyang (2013). Location planning for transit-based evacuation under the risk of service disruptions. Transportation Research Part B: Methodological, 54, 1-16.
3. R.D. Armstrong and Z. Jin (1997). A new strongly polynomial dual network simplex algorithm. Mathematical programming, 78(2), 131-148.
4. T.N. Dhamala and U. Pyakurel (2013). Earliest arrival contraflow problem on series-parallel graphs. International Journal of Operations Research, 10, 1-13.
5. T.N. Dhamala, U. Pyakurel and S. Dempe (2018). A critical survey on the network optimization algorithms for evacuation planning problems. International Journal of Operations Research, 15(3), 101-133.
6. R.C. Dhungana and T.N. Dhamala (2018). FlowLoc Problems on Evacuation Network. In The 11th Triennial Conference of Association of Asia Pacific Operational Research Societies (APORS 2018, August 6-9), 121-123.
7. R.C. Dhungana, U. Pyakurel and T.N. Dhamala (2018). Abstract contraflow models and solution procedures for evacuation planning. Journal of Mathematics Research, 10(4), 89-100.
8. F.R. Ford and D.R. Fulkerson (1958). Constructing maximal dynamic flows from static flows. Operations Research, 6, 419-433.
9. M. Goerigk, K. Deghdak, and P. Heßler (2014). A comprehensive evacuation planning model and genetic solution algorithm. Transportation Research, Part E, 71, 82-97.
10. M. Goerigk, B. Grün and P. Heßler (2014). Combining bus evacuation with location decisions: A branch-and-price approach, Transportation Research Procedia, 2, 783-791.
11. H.W. Hamacher, S. Heller, and B. Rupp (2013), Flow location (FlowLoc) problems: dynamic network flows and location models for evacuation planning. *Annals of Operations Research*, 207(1), 161–180.
12. S. Heller and H.W. Hamacher (2011). The multi-terminal q-FlowLoc problem: a heuristic. *In Lecture Notes in Computer Science, 6701, Proceedings of the International Network Optimization Conference*, 523–528, Berlin, Springer.
13. H. Jia, F. Ordónez and M. Dessouky (2007). A modeling framework for facility location of medical services for large-scale emergencies. *IIE transactions*, 39(1), 41-55.
14. S. Kim, S. Shekhar and M. Min (2008). Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, 20, 1-15.
15. S. Kongsomsaksakul, C. Yang, and A. Chen (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, 6, 4237-4252.
16. E. Köhler, K. Langkau and M. Skutella (2002). Time expanded graphs for flow dependent transit times. *R. Möhring and R. Raman (Eds.): ESA 2002, LNCS 2461, Springer-Verlag*, 599-611.
17. A. Kulshrestha, Y. Lou, and Y. Yin (2014). Pickup locations and bus allocation for transit-based evacuation planning with demand uncertainty. *Journal of Advanced Transportation*, 48(7), 721-733.
18. M. Lin and P. Jaillet (2015). On the quickest flow problem in dynamic networks— a parametric min-cost flow approach. *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1343-1356.
19. M. Ng, J. Park, and S.T. Waller (2010). A hybrid bilevel model for the optimal shelter assignment in emergency evacuations. *Computer-Aided Civil and Infrastructure Engineering*, 25(8), 547-556.
20. J. B. Orlin (2013). Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 765-774.
21. U. Pyakurel (2016). Evacuation planning problem with contraflow approach. PhD Thesis, IOST, Tribhuvan University, Nepal.
22. U. Pyakurel and T.N. Dhamala (2014). Earliest arrival contraflow model for evacuation planning. *Neural, Parallel, and Scientific Computations [CNLS-2013]*, 22, 287-294.
23. U. Pyakurel and T.N. Dhamala (2015). Models and algorithms on contraflow evacuation planning network problems. *International Journal of Operations Research*, 12, 36-46.
24. U. Pyakurel and T.N. Dhamala (2016). Continuous time dynamic contraflow models and algorithms. *Advances in Operations Research - Hindawi*; Article ID 368587, 1-7.
25. U. Pyakurel and T.N. Dhamala (2017a). Evacuation planning by earliest arrival contraflow, *Journal of Industrial and Management Optimization*, 13, 489-503.
26. U. Pyakurel and T.N. Dhamala (2017b). Continuous dynamic contraflow approach for evacuation planning. *Annals of Operations Research*, 253, 573-598.
27. U. Pyakurel, T.N. Dhamala and S. Dempe (2017a). Efficient continuous contraflow algorithms for evacuation planning problems. *Annals of Operations Research (ANOR)*, 254, 335-364.
28. U. Pyakurel, H.H. Nath and T.N. Dhamala (2018). Efficient contraflow algorithms for quickest evacuation planning. *Science China Mathematics*, 61(11), 2079-2100.
29. U. Pyakurel, H.H. Nath and T.N. Dhamala (2018). Partial contraflow with path reversals for evacuation planning. *Annals of Operations Research*, DOI: 10.1007/s10479-018-3031-8.
30. U. Pyakurel, S. Dempe and T.N. Dhamala (2018). Efficient algorithms for flow over time evacuation planning problems with lane reversal strategy. TU Bergakademie Freiberg.
31. U. Pyakurel, H.W. Hamacher and T.N. Dhamala (2014). Generalized maximum dynamic contraflow on lossy network. *International Journal of Operations Research Nepal*, 3, 27-44.
32. S. Rebennack, A. Arulselvan, L. ELEFTERIADOU and P.M. Pardalos (2010). Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, 19, 200-216.
33. B. Rupp (2010). *FlowLoc: Discrete facility locations in flow networks*, Diploma thesis, University of Kaiserslautern, Germany.
34. S. Ruzika, H. Sperber and M. Steiner (2011). Earliest arrival flows on seriesparallel graphs. *Networks*, 57(2), 169-173.
35. M. Saho and M. Shigeno (2017). Cancel-and-tighten algorithm for quickest flow problems. *Network*, 69(2), 179-188.
36. H.D. Sherali, T.B. Carter, and A.G. Hobeika (1991). A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B: Methodological*, 25(6), 439-452.
37. Y. Sheffi (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Englewood Cliffs.
38. M. Skutella (2009). An introduction to network flows over time. In *Research trends in combinatorial optimization*, 451-482.
39. Torres, F. E. (1990). Linearization of mixed-integer products. *Mathematical programming*, 49(1), 427-428.

1Tribhuvan University, Bhaktpur Multiple Campus, Bhaktpur, Nepal, 2,3 Central Department of Mathematics, Tribhuvan University, P.O.Box 13143, Kathmandu, Nepal; 4 TU Bergakademie, Fakultät für Mathematik und Informatik, 09596 Freiberg, Germany; 2 Currently: TU Bergakademie, Fakultät für Mathematik und Informatik, 09596 Freiberg, Germany

E-mail address: 1hari672@gmail.com, 2urmilapyakurel@gmail.com
E-mail address: 3amb.dhamala@daadindia.org, 4dempe@math.tu-freiberg.de