OPTIMIZATION MODELS AND ALGORITHMS FOR EVACUATION PLANNING

A THESIS SUBMITTED TO THE

CENTRAL DEPARTMENT OF MATHEMATICS

INSTITUTE OF SCIENCE AND TECHNOLOGY

TRIBHUVAN UNIVERSITY

NEPAL

FOR THE AWARD OF
DOCTOR OF PHILOSOPHY
IN MATHEMATICS

BY

HARI NANDAN NATH

NOVEMBER 2020

DECLARATION

Thesis entitled **“Opttimizaion Models and Algorithms for Evacuation Planning"** which is being submitted to the Central Department of Mathematics, Institute of Science and Technology (IOST), Tribhuvan University, Nepal for the award of the degree of Doctor of Philosophy (Ph.D.), is a research work carried out by me under the supervision of Prof. Dr. Tanka Nath Dhamala, Central Department of Mathematics, Tribhuvan University and co-supervised by Prof. Dr. Stephan Dempe, Fakultät für Mathematik und Informatik, Technische Universität Bergakademie Freiberg, Germany.

This research is original and has not been submitted earlier in part or full in this or any other form to any university or institute, here or elsewhere, for the award of any degree.

Hari Nandan Nath

RECOMMENDATION

This is to recommend that **Hari Nandan Nath** has carried out research entitled **“Optimization Models and Algorithms for Evacuation Planning"** for the award of Doctor of Philosophy (Ph.D.) in **Mathematics** under our supervision. To our knowledge, this work has not been submitted for any other degree.

He has fulfilled all the requirements laid down by the Institute of Science and Technology (IOST), Tribhuvan Unviersity, Kirtipur for the submission of the thesis for the award of Ph.D. degree.

Dr. Tanka Nath Dhamala
Supervisor
(Professor)
Central Department of Mathematics
Tribhuvan Unviersity
Kirtipur, Kathmandu, Nepal

Dr. Stephan Dempe
Co-Supervisor
(Professor)
Fakultät für Mathematik und Informatik
Technische Universität Bergakademie Freiberg
09596, Freiberg, Germany

November 2020

Phone No. :00977-14331977

TRIBHUVAN UNIVERSITY

CENTRAL DEPARTMENT OF MATHEMATICS

OFFICE OF THE HEAD OF DEPARTMENT

KIRTIPUR, KATHMANDU
NEPAL

Ref. No.:

Date: 06/12/2020

LETTER OF APPROVAL

On the recommendation of **Prof. Dr. Tanka Nath Dhamala** and **Prof. Dr. Stephan Dempe**, this Ph.D. thesis submitted by **Mr. Hari Nandan Nath** entitled **"Optimization Models and Algorithms for Evacuation Planning"** is forwarded by Central Department Research Committee (CDRC) to the Dean, IOST, T.U.

....

Prof. Dr. Tanka Nath Dhamala
(Head)
Central Department of Mathematics
Tribhuvan University
Kirtipur, Kathmandu

ACKNOWLEDGEMENTS

I am grateful to my supervisor Prof. Dr. Tanka Nath Dhamala for invoking my interest in the field of mathematical optimization. This work would not have been possible without his encouragement, inspiration, and guidance. I am also grateful to Prof. Dr. Stephan Dempe who showed me new directions, agreed to become a co-supervisor, and made arrangements for a year-long research stay and study opportunities at Technische Universität Begakademie Freiberg. I am indebted to Dr. Urmila Pyakurel, who is a coauthor in most of the publications related to this thesis, for involving me in her research extensions. I would also like to thank Mrs. Caterina Löschner, secretary at Institut für Numerische Mathematik und Optimierung, TU Bergakademie Freiberg for making my research stay easier there.

I thank University Grants Commission, Bhaktapur Nepal for providing me a Ph.D. Fellowship, Nepal Mathematical Society for NMS fellowship, Bhaktapur Multiple Campus, and Tribhuvan University for granting me a study leave, and the Central Department of Mathematics for arranging a friendly study environment.

I would like to acknowledge DAAD funded GraThO (Graph Theory and Optimization with Applications in Society and Industry) project and its collaborators Prof. Dr. Horst Hamacher, Prof. Dr. Sven O. Krumke, Prof. Dr. Sergio Canoy Jr., Prof. Dr. Ferdinand Jamil including my supervisor Prof. Dhamala, for providing fruitful workshops and research stay supports of three months each, at Mindanao State University-Iligan Institute of Technology, Philippines and Technische Universität Kaiserslautern, Germany.

I am also thankful to GraFA at TU Bergakademie Freiberg for providing me a research stay support via DAAD IPID4all Program, and Alexander von Humboldt Research Group Linkage Program for research-enhancing workshops and support.

Finally, I thank my friend Mr Parishwar Acharya for his initiation to encourage me towards this research area and my wife Pratima Bharati for her unconditional support.

Hari Nandan Nath
November 2020

ABSTRACT

An important strategy, to save a life from natural or human-created disasters, is to evacuate the population from the disastrous zones to safe places. Intelligent evacuation planning requires a carefully designed traffic plan with optimal use of the facilities available.

Reversing the direction of the traffic flow in the appropriate road segments, known as a contraflow approach, has been an important strategy in evacuation planning. To avoid unnecessary arc reversals, we introduce partial contraflow approach and design strongly polynomial algorithms to solve maximum static, maximum dynamic, and quickest flow problems with partial arc reversals, and polynomial-time algorithms to solve maximum static/dynamic abstract partial contraflow problems. Sometimes the travel time on an arc may change when it is reversed. We propose a network transformation that helps convert the existing contraflow algorithms to the ones with orientation-dependent transit times.

To address the dependency of transit times in the flow, we extend the contraflow approach to inflow-dependent transit times and load-dependent transit times. Realizing the NP-hardness of such problems, we propose strongly polynomial $(2 + \epsilon)$-approximation algorithms to solve the corresponding contraflow/partial contraflow problems.

If facilities are to be adjusted on arcs to facilitate evacuation, there may be an increase in the evacuation time. We introduce the quickest FlowLoc model to address such a problem. We prove that the single facility case of the problem can be solved in strongly polynomial time. Proving NP-hardness of the multi-facility case, we propose two heuristics. Taking the case of a Kathmandu road network, the faster heuristic has an average optimality gap of 3.48% and an average running time of 0.17 seconds. The corresponding values for the slower heuristic are 0.18% and 1.02 seconds. The algorithms for quickest FlowLoc problem with arc reversals are also designed.

With an objective to maximize the static/dynamic flows, and minimize quickest flow, the problem of choosing a single shelter location are modeled as MaxStatic, MaxDynamic, Quickest sink location problems respectively. We establish that each of such a problem can be solved in strongly polynomial time with or without arc reversals.

By reversing the direction of the traffic flow towards the sink, a contraflow configuration may obstruct the paths towards the source. We model the problem of maximizing the dynamic contraflow saving a path not exceeding a given length as a mixed binary integer linear programming problem. The analogous problem of minimizing the evacuation time is modeled as a mixed binary integer programming problem with a fractional objective. A linearization strategy is suggested so that the algorithms to solve the mixed integer linear programming problems can be used. The problem of minimizing the path length and maximizing the dynamic contraflow has been modeled as a bicriteria optimization problem. A procedure using ϵ-constrained method is constructed to obtain efficient solutions. The solutions, using available software solvers, considering a road network of Kathmandu city can be obtained within 1 second. We also model the problem of minimizing the path length and evacuation time as a bicriteria problem and construct a procedure to solve it.

We model a path saving model maximizing the dynamic contraflow to optimize a general objective, as a bilevel program. To solve the problem, we replace the lower level problem by Karush-Kuhn-Tucker (KKT) conditions converting it to a single level non-linear binary integer program. We linearize it using a big M method and also suggest a procedure to tune M.

LIST OF ABBREVIATIONS

BPMDC	: Bicriteria path-saving model maximizing dynamic contraflow
BPQC	: Bicriteria path-saving model with quickest contraflow
BPR	: Bureau of Public Roads
GCS	: Generalized cut set
IFDTT	: Inflow-dependent transit times
KKT	: Karush-Kuhn-Tucker
LDTT	: Load dependent transit times
LMSPCF	: Lexicographic maximum static partial contraflow
MDCF	: Maximum dynamic contraflow
MDPCF	: Maximum dynamic partial contraflow
MSCF	: Maximum static contraflow
MSPCF	: Maximum static partial contraflow
QCF	: Quickest contraflow

LIST OF SYMBOLS

$\varnothing$	empty set
2^E	power set of the set E
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^q$	$\{(a_1, \cdots a_q) : a_1, \cdots, a_q \in \mathbb{R}\}$
$\mathbb{R}_{\geq 0}$	set of non-negative real numbers
N	network
V	set of vertices or nodes
E	set of edges or arcs
n	number of nodes $\|V\|$
m	number of arcs $\|E\|$
N'	auxiliary network
$\mathcal{N}$	abstract network
$\mathscr{P}$	set of abstract paths
E_i^-	set arcs with head i
E_i^+	set arcs with tail i
V_i^-	$\{j \in V : (j,i) \in E\}$
V_i^+	$\{j \in V : (i,j) \in E\}$
u	capacity function
τ	transit time function
s	source
t	sink
S	set of sources
T	set of sinks
x	static flow
$\text{excess}_x(i)$	excess of the static flow x at a node i
$v(x)$	value of the flow x
N^x	residual network of N corresponding to flow x
θ	time horizon

X	dynamic flow
$\text{excess}_X(i, \xi)$	excess of the dynamic flow f at a node i and time ξ
$v_\theta(X)$	value of dynamic flow X with time horizon θ
$N^x(\Delta)$	delta residual network
Q	supply at the source
L	set of feasible locations ($L \subseteq E$)
$\nu(i, j)$	number of facilities that can be placed on $(i, j) \in L$
F	set of facilities
σ	size of facilities $\sigma : F \to \mathbb{N}$
γ	allocation function $\gamma : F \to L$
Γ	collection of all possible allocations γ
u_{ij}^{γ}	capacity of arc (i, j) after allocation γ

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

As a living being, one of the basic instincts of a human being is “survival". In spite of developments in science and technology, a significant portion of the population around the world is affected by natural disasters (e.g., earthquakes, tsunamis, hurricanes, tornadoes) and disasters created because of human ignorance, error, intelligence, or intent (e.g, nuclear accidents, fire, poisonous chemical release, terrorism).

The emergency management activities to mitigate the effects of a disaster can, broadly, be categorized as pre-disaster operations, and post-disaster operations (Caunhye, Nie, & Pokharel, 2012). The pre-disaster operations include evacuation from the potential disaster sites to safe shelters, stock pre-positioning, and locating facilities, e.g. shelters, medical centers, facility stores. The post-disaster operations include evacuation from the disaster sites to the shelters, relief distribution from the medical centers or the stores to the disaster sites, transportation of casualties to the medical centers (Figure 1). To carry out these operations with the optimal use of the resources available, the mathematical optimization models available in literature can be broadly categorized into facility location models, evacuation planning models, relief distribution or humanitarian logistic models (Dhamala, Adhikari, Nath, & Pyakurel, 2018).

1.1.1 Evacuation planning models

According to DHS (2004), an evacuation of a region is “organized, phased, and supervised withdrawal, dispersal, or removal of civilians from dangerous, or potentially dangerous areas, and their reception, and care in safe areas". The mathematical models related to evacuation planning focus on finding the optimal use of vehicles and the routes in the complex urban road networks to shift the residents from the danger zones to safe areas. The majority of models assume some kind of vehicle to be used for transferring evacuees to the shelters. In bus-based (also known

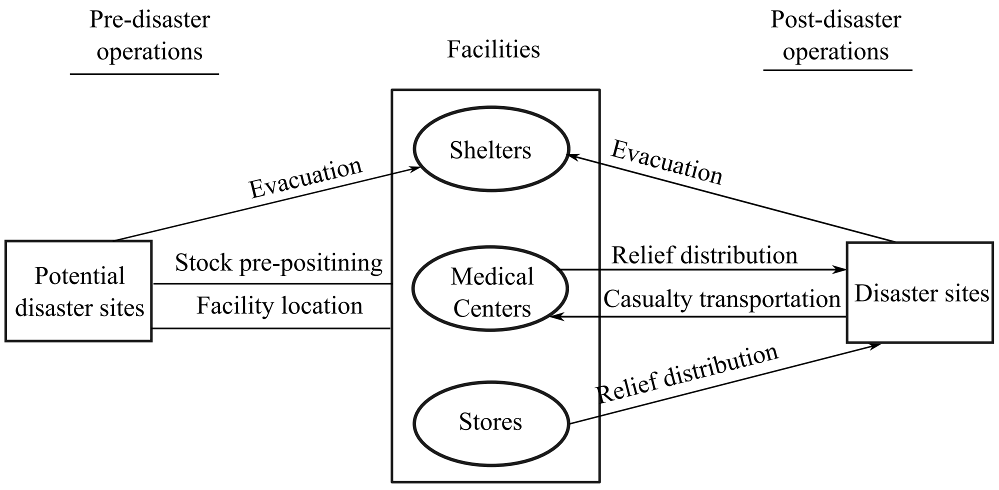


Figure 1: Framework for disaster operations (Caunhye et al., 2012)

as transit-based) models (Bish, 2011; Goerigk, Grün, & Heßler, 2013; Pyakurel, Goerigk, Dhamala, & Hamacher, 2015; Adhikari, Pyakurel, & Dhamala, 2020; Adhikari & Dhamala, 2020), a vehicle can move to and fro between disaster sites and shelters, while in auto-based models, a vehicle does not return to the disaster site. These models represent the road network as a directed network (or graph) with road segments taken as arcs and their intersections as nodes. The disaster sites correspond to sources and the shelters correspond to the sinks.

Among the auto-based models, the maximum flow models focus on rescuing the maximum number of evacuees within a given time horizon, the quickest flow models minimize the time of evacuation of a given number of evacuees. The earliest arrival flow models maximize the number of evacuees at each point of time, while the lexicographic flow models focus on optimizing the flow with a priority order given to the disaster sites and shelters. For the comprehensive study of evacuation planning problems, we refer to Cova and Johnson (2003); Hamacher and Tjandra (2001); Kotsireas, Nagurney, and Pardalos (2015); Akter and Wamba (2019); Dhamala, Pyakurel, and Dempe (2018).

1.1.2 Contraflow approach

In emergency 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 a 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 segments is increased. Contraflow configuration not only increases flow value but also reduces traffic-jam and makes the vehicle-flow smooth. But to identify appropriate directions of the arcs of a network to maximize the flow is a difficult optimization problem, known as a contraflow problem.

Kim, Shekhar, and Min (2008) firstly model the contraflow problem as an integer programming problem, thereby proving its NP-hardness. As finding exact mathematical solutions for general contraflow techniques are costly, they present two greedy and bottleneck heuristics for possible numerical approximate solutions to the quickest contraflow problem. With computational experiments, it has been shown that at least 40% evacuation time can be reduced by reverting at most 30% arcs. In cases, when each node has an associated danger factor, Vogiatzis, Walteros, and Pardalos (2013) present a heuristic algorithm to solve the problem of sending vehicles from the nodes with danger factors to the safe nodes, reversing at most a given number of arcs, with an objective to minimize the number of vehicles that have to spend time on the most endangered nodes. They use the smart clustering of similar nodes to create subgraphs so as to solve a large-scale problem efficiently.

Apart from the heuristic techniques, recent research also focuses on analytical techniques to find exact solutions to the contraflow problems after the introduction of polynomial-time algorithms to solve single-source-single sink maximum contraflow and quickest contraflow problems by Rebennack, Arulselvan, Elefteriadou, and Pardalos (2010). The earliest arrival and the maximum contraflow problems are solved in Dhamala and Pyakurel (2013); Pyakurel (2016) taking time as a discrete parameter. The solution procedures for such problems in a continuous-time setting are described in Pyakurel and Dhamala (2016). Pyakurel and Dhamala (2015) design algorithms to solve the earliest arrival contraflow on single-source-single-sink networks in a pseudo-polynomial time. They also introduce the lexicographic maximum dynamic contraflow problem in which the flow is maximized in a given priority ordering and construct solution algorithms with a polynomial-time complexity. Algorithms for these problems in taking time as a continuous parameter can be found in Pyakurel and Dhamala (2016, 2017a). With the given supplies and demands, the earliest arrival transshipment contraflow problem is modeled in discrete-time and solved on a multi-source network with a polynomial-time algorithm in Pyakurel and Dhamala (2017b). With a zero transit time on each arc, the problem is also solved on a multi-sink network with a polynomial-time complexity. For the multi-terminal network, they present approximation algorithms to solve the earliest arrival transshipment contraflow problem. The discrete-time solutions are extended into the continuous-time solutions in Pyakurel and Dhamala (2017a), and in Pyakurel, Dhamala, and Dempe (2017). The maximum dynamic and the earliest arrival contraflow problems are generalized in Pyakurel, Hamacher, and Dhamala (2014).

Dhungana and Dhamala (2020) consider the problem of maximizing the flow within a given budget considering the cost of arc reversal.

The analytical techniques discussed above use the network flow approach in which a network is represented as a set of nodes and arcs. Recently a formulation of a similar problem with the abstract flow on abstract networks (in which a network is taken to consist of elements and paths (see Section 2.4)) is also gaining attention. Pyakurel et al. (2017) introduce the contraflow technique in abstract networks, present algorithms to solve the maximum static and the maximum dynamic contraflow problems with continuous-time setting and realize that if the minimum dynamic cut capacities on a two-terminal network are symmetric, then the flow value can be increased up to double with the partial contraflow reconfiguration. The models and algorithms for the abstract contraflow problems with discrete-time setting have been investigated in Dhungana, Pyakurel, and Dhamala (2018).

1.1.3 Facility locations in evacuation models

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 the 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 the Large Scale Emergency Medical Service Facility Location Model (LEMS) presented in Jia, Ordóñez, and Dessouky (2007), there is a use of the aforementioned models.

A recent trend in related research incorporates location decisions along with other decisions related to evacuation. The following are some examples.

1. Pick-up location models: An, Cui, Li, and Ouyang (2013) 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 an integrated bus evacuation problem, Goerigk, Grün, and Heßler (2014) choose pick-up locations to minimize the maximum travel time over all the buses. Kulshrestha, Lou, and Yin (2014) use robust optimization to locate pick-up locations when the number of evacuees is uncertain. Nath and Dhamala (2017) propose a model that minimizes the demand-weighted distance between an original node and the nearest pick-up location.

2. Rescue Transfer Location Models: An et al. (2013) formulate a model to locate

rescue transfer locations, where a rescue team departs from the rescue center, rides a vehicle towards the rescue transfer center, and walks to each group to be rescued to provide aid with an objective of minimizing the total expected travel cost.

3. Shelter Location Models: Sherali, Carter, and Hobeika (1991) formulate a location-allocation model to minimize the total vehicle hours, and a discrete median location model to locate shelters for evacuation, while Kongsomsaksakul, Yang, and Chen (2005) use a 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 a combined trip distribution and assignment problem.Ng, Park, and Waller (2010) also use the same approach in which the lower level is a deterministic user equilibrium model as described by Sheffi (1985). In an integrated bus evacuation problem, Goerigk, Grün, and Heßler (2014) choose the shelters to minimize the maximum travel time of all the buses while in a comprehensive evacuation planning, Goerigk, Deghdak, and Heßler (2014) formulate a multi-commodity, multi-criteria problem to minimize the total evacuation time, the risk exposure of evacuees, and the number of shelters that are used.
4. Flow Location (FlowLoc) Models: Rupp (2010); Heller and Hamacher (2011); Hamacher, Heller, and Rupp (2013) combine the location decisions with the flow decisions in a network flow problem observing that the placement of a facility on an arc of a network may result in a reduction of the maximum flow value. Given a set of facilities and a 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 value is minimum.

1.2 Rationale of the Study

A well-planned evacuation is pertinent when a large population is under the threat of some kind of disaster. The overview of literature presented in the last section shows that mathematical modeling for evacuation planning is one of the growing research areas and there is much more to be done for an optimized plan with desired objectives. The contraflow approach has been seen as an important strategy to maximize the number of evacuees and minimize the evacuation duration and the arc reversal strategies in the available literature either reverse the whole road segment or do not reverse it. The whole of the reversed capacities may not have been used. So, it is plausible to design strategies to reverse only the required capacity of the segment. The contraflow models which can be solved analytically do not consider the dependency of transit time in the flow. To develop more realistic models, one needs to consider the dependency of travel time in the direction of the flow and on the amount of flow.

FlowLoc models, available in the literature, to adjust facilities on the road segments focus on maximizing the number of evacuees within a given time horizon. If there is a known number of evacuees, there is a need to develop models that minimize the evacuation time horizon of the evacuees.

The direction of traffic is directed towards the safe places during evacuation. If some facilities are to be moved towards the hazardous areas, especially in the contraflow approach, there may not be a path available. So, it is reasonable to identify the path for the movement of the facilities.

1.3 Objectives

The general objective of this study is to construct mathematical models and develop solution procedures to optimize traffic flow with the adjustment of facilities during an emergency evacuation. The specific objectives are as follows.

- To develop mathematical models to maximize the number of evacuees or minimize evacuation time incorporating facility allocation and facility movement during emergency evacuation
- To design solution algorithms/procedures to solve the formulated models
- To test the computational performance of the algorithms designed

1.4 Structure of the Thesis

The thesis is organized in the following way. Chapter 2 gives the very basic ideas of flows in a network and in an abstract network.

Based on the existing contraflow approach, we develop the partial contraflow approach in Chapter 3 to reverse the direction of the traffic flow up to the required capacity in road segments during an emergency evacuation. The contraflow approach with the transit time on an arc depending on its orientation is also introduced in the chapter.

Chapter 4 extends the contraflow/partial contraflow approach to the models with variable transit time on arcs. We focus, particularly, on the models with inflow-dependent transit time and density dependent-transit time on arcs.

In Chapter 5, we model the problem of optimal allocation of facilities to a set of given arcs with the objective of minimizing the evacuation time. Exact algorithms for single-facility cases and heuristics for multi-facility cases are presented. The problem of identifying optimal sink with an objective to maximize the flow or to minimize the evacuation time is also modeled in the chapter. The corresponding algorithms with arc reversals have also been discussed.

Realizing the need of a path to transport some facilities towards the disaster site, we develop path-saving models to optimize the contraflow in Chapter 6. Mixed binary integer programming models to optimize the contraflow saving a path of a given length bound towards the source have been presented. A bi-criteria approach to minimize the path length and optimizing the contraflow is also described. The problem is also modeled as a bilevel program in which the upper level chooses a path and the lower level maximizes the contraflow saving the path chosen.

Chapter 7 summarizes the thesis and gives further research directions emanated from the presented work.

CHAPTER 2

BASIC NOTIONS

This chapter, briefly, outlines the basic mathematical notions required for the development of the models and algorithms discussed in the work. As we adopt the network flow approach, which is based on mathematical programming (linear programming in most cases) defined on a network and specialized algorithms. The majority of literature considers a network as a directed graph with some attributes on arcs and nodes. An alternative view is an abstract network which consists of a set of elements along with a set of paths, a path being a set of elements with some order.

2.1 Network

A network $N = (V, E)$ consists of nonempty finite sets V of nodes or vertices and E of edges or arcs. Each arc $e \in E$ is associated with a unique ordered pair of nodes in $V \times V$. An arc e associated with $(i, j) \in V \times V$ is said to be directed from i to j with **tail** i, and **head** j. We denote the set of arcs having the node i as the tail by E_i^+, i.e.,

$$E_i^+ = \{e \in E : \text{ tail of } e \text{ is } i\}$$

and the set of arcs having the node i as the head by E_i^-, i.e.,

$$E_i^- = \{e \in E : \text{ head of } e \text{ is } i\}.$$

If there is exactly one arc e with tail i and head j for each $e \in E$, we write $e = (i, j)$, and define

$$V_i^+ = \{j \in V : (i, j) \in E\}$$

and

$$V_i^- = \{j \in V : (j, i) \in E\}.$$

To represent a network diagrammatically, we draw a node as a circle and an arc e

directed from i to j as a curve or a straight line between i and j with an arrow pointing towards j.

Example 2.1. A diagrammatic representation of $N = (V, E)$ with $V = \{s, a, b, t\}$ and

$$E = \{(s, a), (s, b), (a, b), (b, a), (a, t), (b, t)\},$$

is shown in Figure 2.

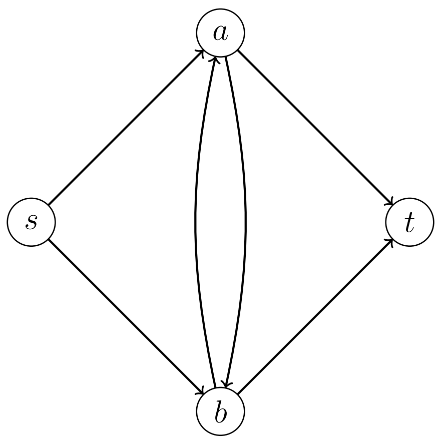


Figure 2: Diagrammatic representaion of N

Moreover,

$$E_a^+ = \{(a, b), (a, t)\}, E_a^- = \{(s, a), (b, a)\}$$

and

$$V_a^+ = \{b, t\}, V_a^- = \{s, b\}$$

Note also that $E_t^+ = E_s^- = \varnothing, V_t^+ = V_s^- = \varnothing$.

Chain and cycle. In a network $N = (V, E)$, let $v_1, \cdots, v_p \in V, e_1, \cdots, e_{p-1} \in E (p \geq 2)$ such that $v_i \neq v_j$ for $i \neq j$. A path from v_1 to v_p, or a v_1–v_p path is a sequence P of nodes and arcs

$$v_1, e_1, v_2, \cdots, e_{p-1}, v_p$$

such that tail of e_i is v_i, head of e_i is v_{i+1} or tail of e_i is v_{i+1}, head of e_i is v_i, $1 \leq i \leq p - 1$. In the path P, if tail of e_i is v_i, head of e_i is v_{i+1} for an admissible i, then it is known as a directed v_1–v_p path or a v_1–v_p chain. The chain P may be represented as v_1–v_2–$\cdots$–v_p. A path which is not a chain is also known as a non-standard chain.

Let $v_1, \cdots, v_p \in V, e_1, \cdots, e_{p-1} \in E (p \geq 2)$ such that $v_1 = v_p$ and $v_i \neq v_j$ for $i \neq j, 2 \leq i \leq p - 1$. The sequence C of nodes and arcs

$$v_1, e_1, v_2, \cdots, e_{p-1}, v_p$$

such that tail of e_i is v_i, head of e_i is v_{i+1} or tail of e_i is v_{i+1}, head of e_i is v_i, $1 \leq i \leq p-1$, is called a cycle. In C, if tail of e_i is v_i, head of e_i is v_{i+1} for all admissible i, then it is known as a directed cycle.

Example 2.2. In Figure 2, The sequence $\{s, (s,a), a, (a,b), b, (b,t), t\}$ is a path which is also a chain or a directed path from s to t (s–t path). The sequence $\{s, (s,a), a, (b,a), b, (b,t), t\}$ is an s–t path but not a chain (i.e. it is a non-standard chain). $\{s, (s,a), a, (a,b), b, (s,b), s\}$ is a cycle which is not directed while $\{a, (a,b), b, (b,a), a\}$ is a directed cycle.

2.2 Static Flows

With a view that some commodity flows from some nodes of a network to some other nodes, a capacity (also known as upper capacity) function $u : E \to \mathbb{R}_{\geq 0}$ is defined on E. The capacity $u(e)$ can be considered as the maximum amount of flow that can arrive at the head of e from its tail (per unit time).

Consider $x : E \to \mathbb{R}_{\geq 0}$. We define the excess of node $i \in V$ by

$$\text{excess}_x(i) = \sum_{e \in E_i^-} x(e) - \sum_{e \in E_i^+} x(e) \tag{1}$$

Given two sets of nodes $S, T \subset V$ (such that $S \cap T = \varnothing$), x is called a **static S–T flow** if it satisfies the **flow conservation constraints**

$$\text{excess}_x(i) = 0,\ \forall i \in V \setminus (S \cup T) \tag{2}$$

and the **capacity constraints**

$$0 \leq x(e) \leq u(e)\ \forall e \in E. \tag{3}$$

The value $v(x)$ of the flow x is defined as

$$v(x) = \sum_{t \in T} \text{excess}_x(t) = \sum_{s \in S} -\text{excess}_x(s) \tag{4}$$

If $\text{excess}_x(i) \geq 0, \forall i \in V$, then x is known as a **pseudoflow**. Each node in S is termed as a **source** node, that in T, a **sink** node, and the remaining nodes are termed as intermediate nodes. If $S = \{s\}, T = \{t\}$, x is called a static s–t flow.

Maximum static flow: For a network with given arc capacities, a static flow x^* is called a maximum static flow, if $v(x^*) \geq v(x)$ for any static flow x defined on the network. In the way the static flow is defined, the maximum static s–t flow problem can be formulated as the following linear program:

$$\max \quad v$$

subject to

$$\begin{aligned} \sum_{e \in E_i^+} x(e) - \sum_{e \in E_i^-} x(e) &= \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -v & \text{if } i = t \end{cases} \\ 0 \leq x(e) &\leq u(e)\, \forall e \in E. \end{aligned}$$

Remark 2.1. If $u(e) \in \mathbb{Z}_{\geq 0}$, for any maximum static flow $x, v(x) \in \mathbb{Z}_{\geq 0}$ and there exists a maximum static flow x^* for which $x^*(e) \in \mathbb{Z}_{\geq 0}$ for each $e \in E$.

Remark 2.2. To find the maximum static S–T flow, we can add two extra nodes s^*, t^* in V and construct arcs $(s^*, s), (t, t^*)$, with infinite capacity, for each $s \in S, t \in T$ and find maximum static s^*–t^* flow.

Static *b*-flow: Given function $b : E \to \mathbb{R}$ and a lower capacity function $l : E \to \mathbb{R}_{\geq 0}$, x is called a b-flow if

$$\text{excess}_x(i) = b(i), \forall i \in V \tag{5}$$

and

$$l(e) \leq x(e) \leq u(e) \tag{6}$$

where $l(e)$ denotes the minimum amount of flow (per unit time) that must arrive at the head e from its tail. If $b(i) = 0\ \forall i \in V$, then x is called a **circulation**.

Remark 2.3. A necessary condition for a b-flow to exist is that $\sum_{i \in V} b(i) = 0$.

Remark 2.4. Any static s–t flow x is a b-flow with $b(i) = 0\ \forall i \neq s, t$, $b(s) = -v(x)$ and $b(t) = v(x)$.

Flow decomposition: Let P be a chain (a directed path) in N with a starting vertex $s(P)$ and end vertex $t(P)$, a **chain flow** x^P of value $\delta > 0$ on P is a static $s(P)$– $t(P)$ flow such that

$$x^P(e) = \begin{cases} \delta & \text{if } e \in P \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

If C is a directed cycle in N, a **cycle flow** x^C with value $\delta > 0$ on C is a circulation such that

$$x^C(e) = \begin{cases} \delta & \text{if } e \in C \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

We can decompose any b-flow into chain and cycle flows. Let $\mathcal{P}$ be the set of all chains and $\mathcal{C}$ be the set of all cycles in N. A b-flow x can be decomposed into at most $|V|+|E|$ (positive) flows on chains and directed cycles in such that

(i) $x(e) = \sum_{P \in \mathcal{P}: e \in P} x^P(e) + \sum_{C \in \mathcal{C}: e \in C} x^C(e)$.

(ii) Every directed path with a positive flow connects a vertex i with $b(i) < 0$ to a vertex j with $b(j) > 0$.

(iii) There are at most $|E|$ circulations in the decomposition.

Minimum cost flow: Given a cost function $c : E \to \mathbb{R}$, the cost of the flow x is given by

$$\sum_{e \in E} c(e)x(e) \tag{9}$$

where $c(e)$ denotes the cost per unit flow for the arc e.

A b-flow x that minimizes the cost (9) is called the minimum cost flow. Thus the minimum cost flow problem can be formulated as the following linear program.

$$\min \quad \sum_{e \in E} c(e)x(e)$$

subject to

$$\begin{aligned} \sum_{e \in E_i^+} x(e) - \sum_{e \in E_i^-} x(e) &= -b(i)\ \forall i \in V \\ l(e) \ \leq x(e) &\leq u(e)\ \forall e \in E \end{aligned}$$

where $\sum_{i \in V} b(i) = 0$ for feasibility. In the above formulation $b(i)$ denotes the demand at each node i. A node i is called a demand node, a supply node, or an intermediate node according as $b(i) > 0$, $b(i) < 0$ or $b(i) = 0$. The minimum cost flow problem identifies the b-flow x that sends the given amount of supply from supply nodes to meet the demand at the demand nodes.

Residual network: Most of the algorithms to find maximum flow or minimum cost flow make use of a network called residual network, the construction of which depends on the static flow. Given a static flow x, the residual network $N^x = (V, E^x)$ contains the same node set V. Associated with each $e \in E^x$, is the residual capacity $u^x(e)$, and cost $c^x(e)$. For each $e \in E$,

- if $x(e) < u(e)$, there is an arc $+e \in E^x$ with head of $+e$ as head of e, tail of $+e$ as tail of e, $u^x(+e) = u(e) - x(e)$, and $c^x(+e) = c(e)$,
- if $x(e) > 0$, there is an arc $-e \in E^x$ with head of $-e$ as tail of e, tail of $-e$ as head of e, $u^x(-e) = x(e) - l(e)$, and $c^x(-e) = -c(e)$.

2.3 Dynamic Flows

Let $\tau : E \to \mathbb{R}_{\geq 0}$ be an arc transit time function. Given a time horizon $\theta \geq 0$, a dynamic flow (or a flow over time) $X = (X_e)_{e \in E}$, where $X_e : [0, \theta) \to \mathbb{R}_{\geq 0}$, is a Lebesgue measurable function such that

$$X_e(\xi) = 0 \text{ for } \xi \geq \theta - \tau(e) \tag{10}$$

where $X_e(\xi)$ represents the rate of flow that enters the tail of e at time ξ and reaches its head at time $\xi + \tau(e)$. So the outflow rate at the head of e at time $\xi + \tau(e)$ is $X_e(\xi)$, and the outflow rate at any time ξ' at the head of e is $X_e(\xi' - \tau(e))$.

We define the excess of the node i at time ξ as

$$\text{excess}_X(i, \xi) = \sum_{e \in E_i^-} \int_0^{\xi - \tau(e)} X_e(\zeta) d\zeta - \sum_{e \in E_i^+} \int_0^{\xi} X_e(\zeta) d\zeta \tag{11}$$

which is the net amount of flow that enters the node i up to time ξ.

Given two distinct nodes $s, t \in V$, a flow over time f is called an s-t flow over time if

$$\begin{aligned} \text{excess}_X(i, \xi) &\geq 0, \forall i \in V \setminus \{s\}, \xi \in [0, \theta) && (12) \\ \text{excess}_X(i, \theta) &= 0, \forall i \in V \setminus \{s, t\} && (13) \\ X_e(\xi) &\leq u(e), \forall e \in E, \xi \in [0, \theta) && (14) \end{aligned}$$

The value of the s–t flow over time is

$$v_\theta(X) = \text{excess}_X(t, \theta) \tag{15}$$

Constraints (12) and (13) allow to store flow at intermediate nodes for some time, as long as it has left the node again before the time horizon is over.

Example 2.3. Figure 3 shows a network N with $V = \{s, i, t\}, E = \{e_1 = (s, i), e_2 = (i, t)\}$ with $u(e_1) = 2, u(e_2) = 1$ and $\tau(e_1) = 1, \tau(e_2) = 2$. For a time horizon of $\theta = 5$,

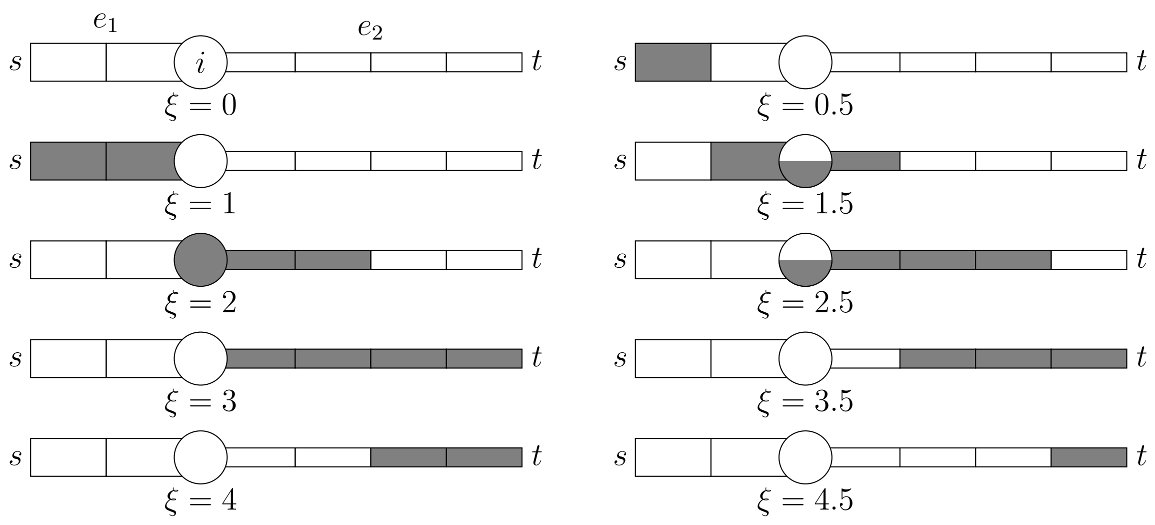


Figure 3: Dynamic s–t flow

consider f defined by

$$X_{e_1}(\xi) = \begin{cases} 2, & 0 \le \xi < 1 \\ 0, & \text{otherwise} \end{cases}$$

and

$$X_{e_2}(\xi) = \begin{cases} 1, & 1 \le \xi < 3 \\ 0, & \text{otherwise.} \end{cases}$$

One can easily verify that f defines a dynamic flow. Note that

$$\text{excess}_X(i, \xi) = \int_0^{\xi-1} X_{e_1}(\zeta)d\zeta - \int_0^{\xi} X_{e_2}(\zeta)d\zeta.$$

For example,

$$\begin{aligned} \text{excess}_X(i, 2.5) &= \int_0^{1.5} X_{e_1}(\zeta)d\zeta - \int_0^{2.5} X_{e_2}(\zeta)d\zeta \\ &= \int_0^{1} 2d\zeta - \int_1^{2.5} 1d\zeta \\ &= 0.5. \end{aligned}$$

This shows the storage of flow at i which, however, is cleared at soon $\xi = 3$. The value of the dynamic flow is

$$v_5(X) = \text{excess}_X(t, 5) = \int_0^3 X_{e_2}(\zeta)d\zeta = \int_1^3 1d\zeta = 2.$$

Temporally repeated flow: Given a feasible static flow x and a time horizon θ, a flow decomposition of x gives a set of paths $\mathcal{P}$ with flow x^P for each $P \in \mathcal{P}$. If a flow is sent along P at a constant rate x^P from the source during the time interval $[0, \theta - \tau(P))$, we can obtain a flow over time with time horizon θ, where $\tau(P) = \sum_{e \in P} \tau(e)$ is the travel time on path P. For a node i in an s–t path P, let P_{si}, P_{it} denote the subpaths of P from s to i and from i to t, and

$$\mathcal{P}_e(\xi) = \{P \in \mathcal{P} : e \in P \text{ and } \tau(P_{si}) \leq \xi \text{ and } \tau(P_{it}) < \theta - \xi \text{ where } i \text{ is tail of } e\}.$$

Then a temporally repeated flow f is obtained as described in the following equation

$$X_e(\xi) = \sum_{P \in \mathcal{P}_e(\xi)} x^P, \ \forall e \in E, \xi \in [0, \theta) \tag{16}$$

The temporally repeated flow described in (16) is a feasible dynamic s–t flow, with strict equality in the constraints (12), i.e. it does not allow waiting in the intermediate nodes (Skutella, 2009).

Example 2.4. Consider the network in Example 2.3 once again. Let us take a static flow x defined by $x(e_1) = x_{e_2} = 1$. The only s–t path P is s–i–t with $\tau(P_{st}) = 3, \tau(P_{si}) = 1, \tau(P_{it}) = 2$. Taking $\theta = 5$, for the temporally repeated dynamic flow X^1 constructed from x,

$X^1_{e_1}(\xi) = 1$ for $0 \leq \xi$ and $3 < 5 - \xi$, i.e.

$$X^1_{e_1}(\xi) = \begin{cases} 1 & \text{for } 0 \leq \xi < 2 \\ 0 & \text{otherwise} \end{cases}$$

and $X^1_{e_2}(\xi) = 1$ for $1 \leq \xi$ and $2 < 5 - \xi$, i.e.

$$X^1_{e_2}(\xi) = \begin{cases} 1 & \text{for } 1 \leq \xi < 3 \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that the value of X^1 is 2 which is equal to that of X in Example 2.3.

Discrete dynamic flow: For a positive integer θ, discretizing the time interval $[0, \theta)$ into the time steps $0, 1, \cdots, \theta - 1$, each corresponding to $[0, 1), [1, 2), \cdots [\theta - 1, \theta)$, if $X_e(\xi)$ represents the amount of flow that enters the tail of e at time ξ and reaches its head at time $\xi + \tau(e)$, the excess of the node i at time $\xi \in \{0, 1, \cdots, \theta - 1\}$ is

$$\text{excess}_X(i, \xi) = \sum_{e \in E_i^-} \sum_{\zeta=0}^{\xi - \tau(e)} X_e(\zeta) - \sum_{e \in E_i^+} \sum_{\zeta=0}^{\xi} X_e(\zeta). \tag{17}$$

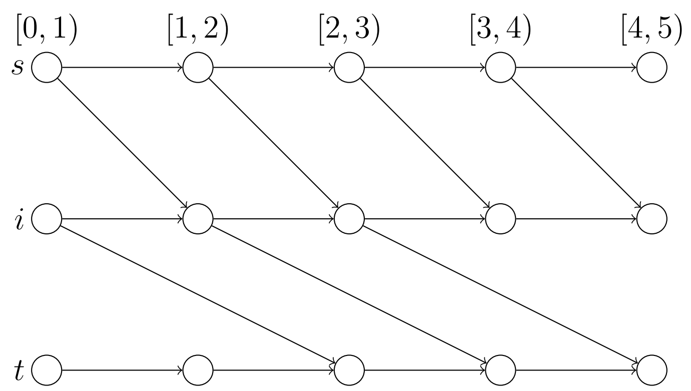


Figure 4: Time-expanded network of the network in Figure 3

Such a flow f is known as discrete dynamic flow. Using the concept of natural transformations, Fleischer and Tardos (1998) show the equivalence between the two problems so that the solution procedures of a problem in continuous time version can be carried to the solution procedure of the corresponding problem in the discrete version, and vice versa.

Time-expanded network: Given an integral time horizon θ, the time-expanded network of N, denoted by N^θ is constructed as follows. For each $[\zeta, \zeta + 1), \zeta \in \{0, 1, \cdots, \theta - 1\}$ and a node $i \in V$, there is a node i_ζ in N^θ. For each $e \in E$ with a tail i and head j, there is an arc, with capacity $u(e)$, directed from i_ζ to $j_{\zeta+\tau(e)}$ if $\zeta + \tau(e) < \theta$. There is an arc in N^θ from i_ζ to $i_{\zeta+1}$ with infinite capacity if the storage of the flow is allowed. See Figure 4 for the time-expanded network of the network given in Example 2.3.

Maximum dynamic flow: Given a network N, and a time horizon $\theta \geq 0$, an s–t flow over time X^* is called a maximum dynamic flow (or a maximum flow over time) if $v_\theta(X^*) \geq v_\theta(X)$ for any dynamic flow X.

Quickest flow: Given a supply Q at the source s, the dynamic flow X of value Q with minimum time horizon θ is called the quickest flow.

2.4 Abstract Network and Abstract Flow

An abstract network $\mathcal{N} = (\mathcal{E}, \mathscr{P})$ consists of a ground set $\mathcal{E}$ of elements and a family of paths $\mathscr{P} \subseteq 2^{\mathcal{E}}$. There is an order $<^P$ of elements defined in each $P \in \mathscr{P}$, and the following **switching property** is satisfied:
$\forall P, Q \in \mathscr{P}, e \in P \cap Q, \exists P \times^e Q \in \mathscr{P}$, such that

$$P \times^e Q \subseteq \{p \in P : p \leq^P e\} \cup \{q \in Q : e \leq^P q\}.$$

For $e_1, e_2 \in P$, if $e_1 <^P e_2$, then we say that e_1 is before e_2 (or equivalently, e_2 is after e_1) in P.

The network considered in the previous sections is often termed as a classical network to make a distinction with the abstract network. In a classical network, if paths P and Q intersect a path R, there must be a path starting from the beginning of P and terminating at the end of Q. This may not be true in abstract networks. The following example illustrates this fact.

Example 2.5 (Kappmeier, Matuschke, and Peis (2014))**.** Consider $\mathcal{N} = (\mathcal{E}, \mathscr{P})$ with $\mathcal{E} = \{1,2,3,4,a,b,c,d\}$, $\mathscr{P} = \{P_1 = \{1,2,3,4\}, P_2 = \{a,2,c\}, P_3 = \{b,3,d\}, P_4 = \{1,c\}, P_5 = \{1,d\}, P_6 = \{a,4\}, P_7 = \{b,4\}\}$. One can verify that $\mathcal{N}$ is an abstract network. Although P_2 and P_3 intersect the path P_1, there is no path that starts with a and ends with d.

As in the case of (classical) networks, we define a capacity function $u : \mathcal{E} \to \mathbb{R}_{\geq 0}$, and a transit time function $\tau : \mathcal{E} \to \mathbb{R}_{\geq 0}$. A **static abstract flow** $x : \mathscr{P} \to \mathbb{R}_{\geq 0}$ is a function that satisfies the capacity constraint

$$\sum_{P \in \mathscr{P} : e \in P} x(P) \leq u(e), \forall e \in \mathcal{E}$$

In this way, we can formulate the maximum static abstract flow problem as:

$$\max \quad \sum_{P \in \mathscr{P}} x(P)$$

subject to

$$\begin{aligned} \sum_{P \in \mathscr{P} : e \in P} x(P) &\leq u(e), \forall e \in \mathcal{E} \\ x(P) &\geq 0, \forall P \in \mathscr{P} \end{aligned}$$

For more details, we refer to McCormick (1996).

We denote the set $\{p \in P : p <^P e\}$ by (P, e), which can be regarded as the set of elements of P that are before e. Given an integral time horizon θ, the intervals $[0,1), \cdots, [\theta - 1, \theta)$ are identified with the set of their starting times

$$\mathcal{T} = \{0, 1, \cdots, \theta - 1\}.$$

For $\xi \in \mathcal{T}$, we define a temporal path P_ξ by

$$P_\xi = \left\{ (e, \zeta) \in \mathcal{E} \times \mathcal{T} : e \in P, \zeta = \xi + \sum_{p \in (P,e)} \tau(p). \right\}$$

The set of all temporal paths is:

$$\mathscr{P}_\mathcal{T} = \left\{ P_\xi : P \in \mathscr{P}, \xi \in \mathcal{T}, \xi + \sum_{p \in P} \tau(p) < \theta. \right\}$$

A **dynamic abstract flow** or an **abstract flow over time** is $X : \mathscr{P}_\mathcal{T} \to \mathbb{R}_{\geq 0}$ such that the capacity of every element at every point in time is respected. Thus the maximum dynamic abstract flow problem can be formulated as

$$\max \quad \sum_{P_\xi \in \mathscr{P}_\mathcal{T}} X(P_\xi)$$

subject to

$$\begin{aligned} \sum_{P_\xi \in \mathscr{P}_\mathcal{T} : (e,\xi) \in P_\xi} X(P_\xi) &\leq u(e), \forall e \in E, \xi \in \mathcal{T} \\ X(P_\xi) &\geq 0, \forall P_\xi \in \mathscr{P}_\mathcal{T} \end{aligned}$$

Given a static abstract flow x and a time horizon θ, we can construct a dynamic abstract flow by setting

$$X(P_\xi) = x(P), \forall P \in \mathscr{P} \text{ and } 0 \leq \xi < \theta - \sum_{e \in P} \tau(e),$$

known as a **temporally repeated abstract flow** (Kappmeier, 2015).

CHAPTER 3

REVERSING LANES FOR EVACUATION

3.1 Introduction

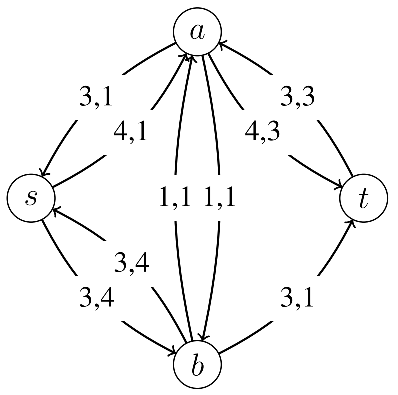
To model the traffic flow in urban road networks using the network flow approach, the road segments are taken as arcs and their intersections are taken as nodes of the network. The direction of the traffic flow in a road segment is taken as the direction of the corresponding arc. To optimize the flow during emergency evacuation, we take hazardous areas as sources and the safe areas, where evacuees are to be transported, as sinks. The corresponding network is often termed as an evacuation network.

Let (V, E) be an evacuation network with source nodes S, and sink nodes T (such that $S \cap T = \varnothing$). Assuming that there are at most two arcs between any two nodes, we denote the arc e directed from a node i to a node j by (i, j). If $u, \tau : E \to \mathbb{R}_{\geq 0}$, are the capacity, and the travel time functions respectively, then we also write $u(i, j) = u_{ij}$, and $\tau(i, j) = \tau_{ij}$ which represent the maximum number of traffic flow units that can travel per unit time and the time taken by the flow to travel, respectively, from i to j. Such a network is denoted by $N = (V, E, u, \tau, S, T)$. If $S = \{s\}, T = \{t\}$, we denote the network as $N = (V, E, u, \tau, s, t)$. If there is one source and one sink, the network is termed as a two-terminal network, otherwise, it is termed multi-terminal. Further, we assume that $|V| = n, |E| = m$.

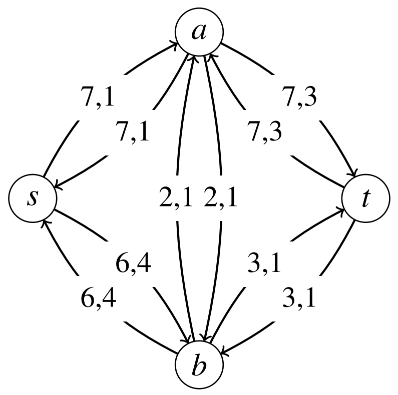
To model the problems using an abstract network $\mathcal{N}$, we can take the nodes or arcs of N as the elements and the path system as the abstract paths (see Section 3.5).

3.2 Contraflow Models

In evacuation situations, if the direction of the usual traffic flow is reversed in appropriate road segments, then the amount of the traffic flow towards the safe areas can be increased. As a result, there is a decrease in the evacuation time of a given number of evacuees.



(a) Network N



(b) Auxiliary network N'

Figure 5: Auxiliary network construction

Given an evacuation network $N = (V, A, u, \tau, s, t)$, contraflow models seek to identify a set of arcs $R \subseteq E$ so that each $(i, j) \in R$ can be replaced by (j, i) to optimize the flow. One of the important strategies to solve some of such problems is to solve the corresponding flow problem in a modified network known as auxiliary network.

Definition 3.1 (Auxiliary network)**.** The auxiliary network of $N = (V, E, u, \tau, s, t)$ with $\tau_{ij} = \tau_{ji}$ for all $(i, j) \in E$ if $(j, i) \in E$, is defined as the network $N' = (V, E', u', \tau', s, t)$ where

$$E' = \{(i, j) : (i, j) \in E \text{ or } (j, i) \in E\},$$

the capacity function $u' : E' \to \mathbb{R}_{\geq 0}$ is given by

$$u'_{ij} = \begin{cases} u_{ij} & \text{if } (j,i) \notin E \\ u_{ji} & \text{if } (i,j) \notin E \\ u_{ij} + u_{ji} & \text{otherwise} \end{cases}$$

and the travel time function $\tau' : E \to \mathbb{R}_{\geq 0}$ is given by

$$\tau_{ij} = \begin{cases} \tau_{ij} & \text{if } (i,j) \in E \\ \tau_{ji} & \text{otherwise.} \end{cases}$$

This concept is depicted in Figure 5. Each arc label represents the capacity and the travel time associated with the corresponding arc.

3.2.1 Maximum static/dynamic contraflow

Definition 3.2 (Maximum static contraflow)**.** Given an evacuation network $N = (V, E, u, \tau, s, t)$, the maximum static contraflow (MSCF) is a maximum static flow reversing the necessary arcs in E.

Definition 3.3 (Maximum dynamic contraflow)**.** Given $N = (V, E, u, \tau, s, t)$ with a time horizon θ, the maximum dynamic contraflow (MDCF) is the maximum dynamic flow reversing the necessary arcs in E at time zero.

Rebennack et al. (2010) developed procedures to find the maximum amount of static flow in two-, multi-terminal networks, and dynamic flow in two-terminal networks, if the arcs can be reversed. To solve the MSCF problem, they solve the maximum static flow problem in the corresponding auxiliary network, find the chain and cycle decomposition of the flow, remove the positive flow in cycles, and reverse such an arc of the original network if the flow in the opposite arc of the auxiliary network exceeds the capacity of the corresponding arc in the original network. To solve the MDCF problem, a similar procedure is followed by finding the static flow corresponding to the temporally repeated maximum dynamic flow in the auxiliary network.

3.2.2 Quickest contraflow

Definition 3.4 (Quickest contraflow)**.** Let $N = (V, E, u, \tau, s, t)$ be an evacuation network with a supply Q at the s. A dynamic flow of value Q with the minimum time horizon, reversing the necessary arcs in E at time zero, is known as the quickest contraflow (QCF).

To solve the QCF problem in the similar way the MDCF problem is solved, one needs to find the quickest flow in the temporally repeated form. Some ways to find such a flow are briefed below.

Searching algorithms: Let f be a maximum dynamic s–t flow with time horizon θ. Then $v_\theta(f)$, the value of f increases as θ increases. Using this property, Burkard, Dlaska, and Klinz (1993) develop various algorithms to find the quickest flow. The main idea is to start with an interval $[\theta_l, \theta_u]$ such that $v_{\theta_l}(f) < Q < v_{\theta_u}(f)$, and search for minimum θ^* such that $v_{\theta^*}(f) \geq Q$. Their algorithms require multiple calls of solving a minimum-cost circulation problem. The strongly polynomial algorithm suggested by them has time-complexity of $O(m^2 \log^3 n(m + n \log n))$.

Cost scaling algorithm: Using the fact that the temporally repeated maximum dynamic flow with a time horizon θ can be obtained by finding the static flow that maximizes $\theta v(x) - \sum_{(i,j)\in E} \tau_{ij} x_{ij}$ (Ford & Fulkerson, 1958; Fleischer & Tardos, 1998), the following result is useful to compute the quickest flow.

Theorem 3.1 (Lin and Jaillet (2015))**.** *Given an evacuation network $N = (V, E, u, \tau, s, t)$ with a supply Q at s, the quickest flow problem can be formulated as the fractional programming problem*

$$\min \quad \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v} \tag{18}$$

subject to

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} \quad = \quad \begin{cases} v & \textit{if } i = s \\ 0 & \textit{if } i \in V \setminus \{s,t\} \\ -v & \textit{if } i = t \end{cases} \tag{18a}$$

$$0 \quad \leq x_{ij} \quad \leq \quad u_{ij}\ \forall (i,j) \in E \tag{18b}$$

If v is fixed in the problem stated in (18) becomes a minimum cost flow problem with supply Q at s and demand Q at t. With this observation, they derive the following optimality conditions:

i) x is a minimum cost flow,

ii) $d_{st} \geq \frac{Q+\sum_{(i,j)\in E}\tau_{ij}x_{ij}}{v} \geq -d_{ts}$,

where d_{st} and d_{ts} are the lengths of the shortest s–t path, and t–s path in the residual network N^x.

Based on these optimality conditions, (Lin & Jaillet, 2015) propose a cost scaling algorithm, which is summarized below.

Given $N = (V, E, u, \tau, s, t)$ with a supply Q at s, let x be a static flow with value v. Node potentials π are introduced and the reduced cost $c_e = \pi_j - \pi_i + \tau_e$ is calculated for each arc ($e \in N_x$ with tail i, and head j, where N^x is the residual network corresponding to x. When $c_e > -\epsilon, \forall e \in N^x$, the obtained flow x is called ϵ-optimal. The following are the steps of the algorithm, in brief.

1. Initialize: $\pi_i = 0, \forall i \in V, x_e = 0, \forall e \in E$, and $\epsilon = C = \max_{e\in A}\{\tau_e\}$.

2. Refine: The 2ϵ-optimal flow is modified to an ϵ-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 potentials if required.

3. Reduce Gap: Set extra flow at s and push the admissible flow ultimately to t with arcs in N^x, and relabel the potential of nodes if required to reduce the gap between $\theta = [Q + \sum_{e\in E} \tau_e x_e]/v$ and $\pi_s - \pi_t$ by at least $7n\epsilon$.

After Step 3, ϵ is scaled by $\frac{1}{2}$, and Steps 2 and 3 are repeated unless $\epsilon < \frac{1}{8n}$.

4. Saturate: If θ, obtained from the above-mentioned scaling phases, is more than the time (cost) in a shortest simple path from s to t in the residual network N^x, the

flow is saturated by sending maximum flow from s to t in a subnetwork formed only by those arcs which are on some shortest path from s to t in N^x.

The time complexity of the cost-scaling algorithm is $O(n^3 \log(nC))$.

Cancel-and-tighten algorithm: 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 i, compute the maximum length $h(i)$ from nodes with no entering admissible arc. Replace π_i by $\pi_i + \frac{\epsilon}{n}h(i)$ and reduce ϵ to $(1-\frac{1}{n})\epsilon$.

The Cancel Steps and the Tighten Steps are repeated iteratively until ϵ reduces to $\epsilon/2$. Then the Reduce Gap step reduces the gap between $\theta = [Q + \sum_{e \in E} \tau_e x_e]/v$ and $\pi_s - \pi_t$ by at least $(3n+1)\epsilon$.

The above-mentioned steps are performed until ϵ becomes smaller than $1/(4n)$, and finally the Saturate step is performed as in the cost scaling algorithm. The complexity of this algorithm is $O(nm^2 \log^2 n)$. For details of the algorithms, we refer to Lin and Jaillet (2015) and Saho and Shigeno (2017).

Using a quickest flow algorithm as a subroutine, we construct Algorithm 1 to solve the quickest contraflow problem, and prove its correctness and discuss the complexity. The results are based on our publication Pyakurel, Nath, and Dhamala (2018).

Algorithm 1: Quickest contraflow algorithm

```
Input  : Evacuation network $N = (V, E, u, \tau, s, t)$ with a supply $Q$ at $s$
Output: Quickest contraflow in $N$
1 Construct the auxiliary network $N' = (V, E', u', \tau', s, t)$.
2 Find a static flow $x$ and the time horizon $\theta^*$ corresponding to a temporally repeated
    quickest flow in $N'$ with supply $Q$ at $s$.
3 Decompose $x$ obtained in Step 2 into chain and cycle flows. Update $x$ by removing
    cycle flows.
4 Reverse $(i,j) \in E$ iff $(j,i) \in E$ and $x_{ji} > u_{ji}$ or $(j,i) \notin E$ and $x_{ji} > 0$. Quickest
    flow in reconfigured $N$ is the temporally repeated flow corresponding to by $x$ with
    the time horizon $\theta^*$.
```

Theorem 3.2. *Algorithm 1 solves the QCF problem optimally.*

Proof. First, we will prove that the algorithm guarantees a feasible dynamic flow in the transformed network after arc reversal. For this we have to prove that the static flow defined by x in the transformed network is feasible. Since removal of cycle flows in

Step 3 assures x to be positive either in (i, j) or in (j, i) but not in both, and x does not exceed the capacity of the corresponding arc even after the arc reversal (because after the reversal of (j, i), the capacity of (i, j) becomes $u_{ij} + u_{ji}$). Next, we have to show that the flow is optimal. Since we solve the quickest flow problem in the auxiliary network N', the flow is optimal in N'. Since the removal of a positive cycle flow retains the value of the flow and does not increase the total cost of the flow, the updated flow is still optimal in N' and hence it is optimal in N after the arc reversals. □

Application of the cost-scaling algorithm or the cancel-and-tighten algorithm, discussed above, in Step 2, enables the solution of QCF problem, respectively, within a polynomial and a strongly polynomial time-complexity of a minimum cost flow problem. The following theorem gives the formal proof of the strongly polynomial time-complexity.

Theorem 3.3. *The QCF problem can be solved in $O(nm^2 \log^2 n)$ time.*

Proof. Consider Algorithm 1 to solve the problem. Step 1 of the algorithm can be performed in $O(m)$ time. With the application of the cancel-and-tighten algorithm of the quickest flow Step 2 takes $O(nm^2 \log^2 n)$ time. Flow decomposition in Step 3 takes $O(mn)$ time (Ahuja, Magnanti, & Orlin, 1993). Step 4 can also be done in $O(m)$ time. This shows that the complexity of the algorithm is dominated by the complexity of Step 2 which is $O(nm^2 \log^2 n)$. □

3.2.3 Computational results

To compare the tests before and after contraflow configuration in evacuation planning, we consider a situation where evacuees are to be evacuated from a single source (e.g. evacuees at a place of some event or evacuees gathered at a place after earthquake etc.) to a single sink (e.g. a bus park or a train station, etc.). For computational experiments, we construct a network of 24 nodes and 76 arcs (38 two way links) and choose the capacities of arcs between 1 to 3 cars per second with travel time between 5 minutes to 10 minutes (the network topology of the considered network is given in Appendix A.1). The results are published in Pyakurel et al. (2018).

Taking the number of evacuee-cars from 500 to 50000 (with a gap of 500), we find that the difference between the quickest time before and after contraflow configuration increases with the increase in the number of cars (Figure 6).

In the same set-up, the value of the static flow rate, $v(x)$, after contraflow is found higher than that before contraflow (Figure 7), maintaining a constant difference after certain level of number of cars (10000 in this case). With only 29% of the link reversals,

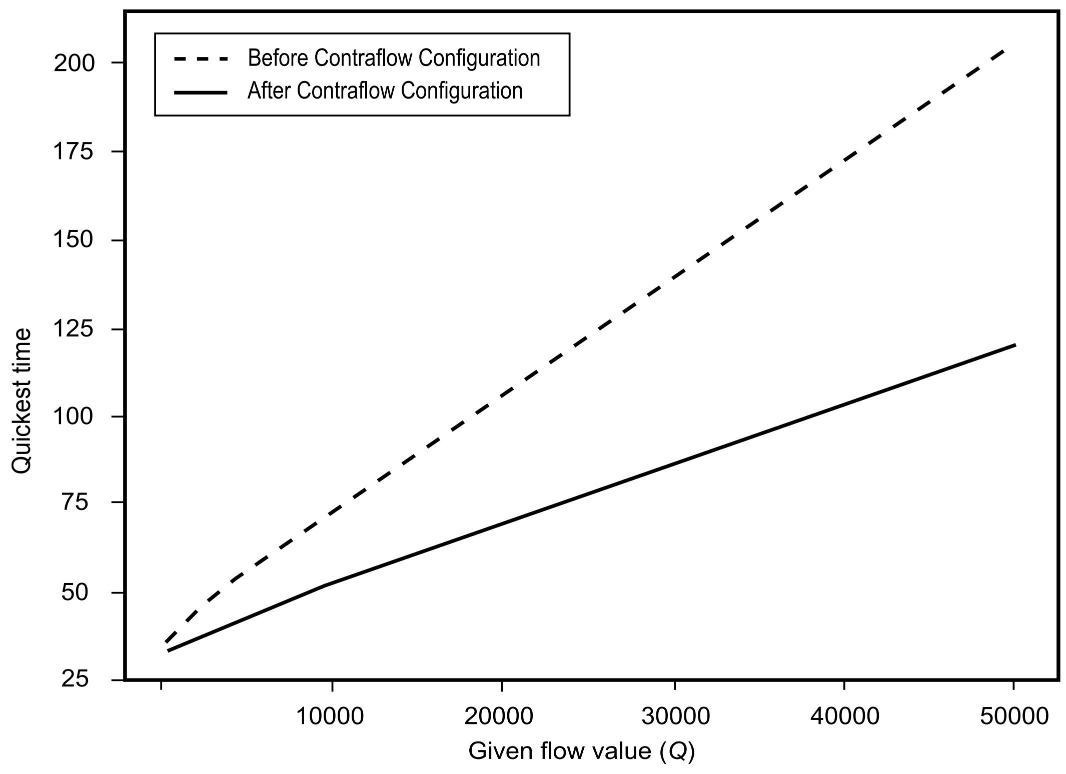


Figure 6: Quickest time comparison

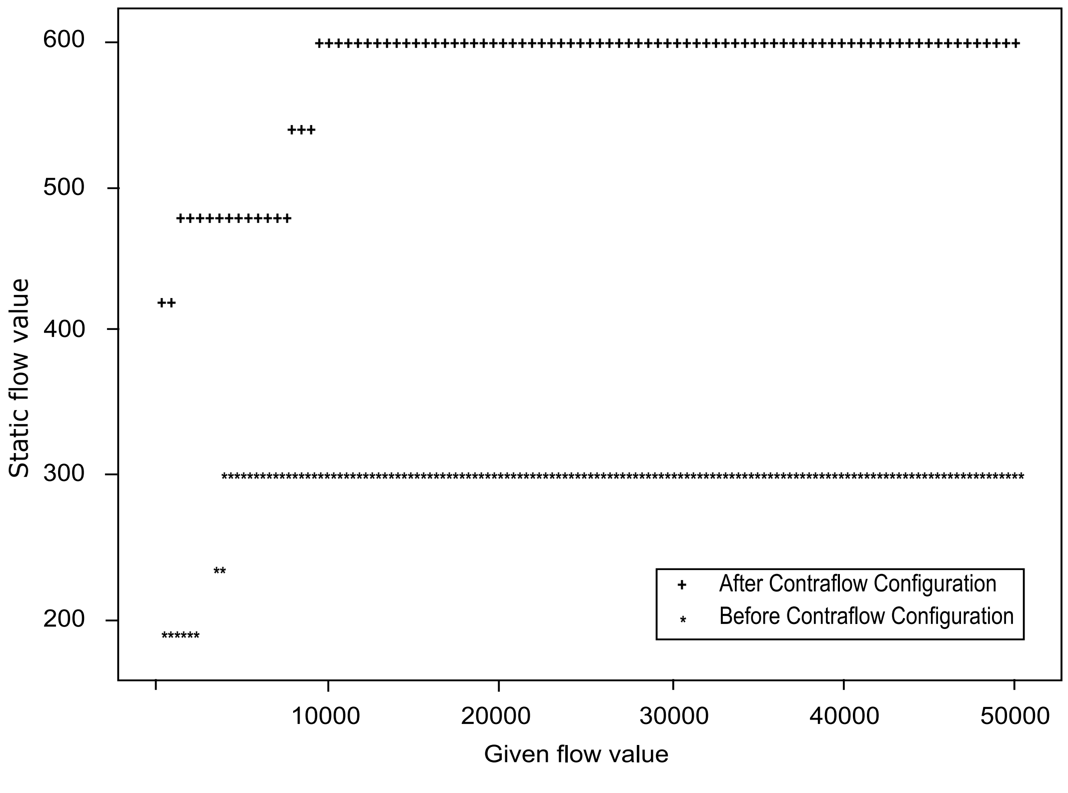


Figure 7: Static flow rates corresponding to the quickest flow

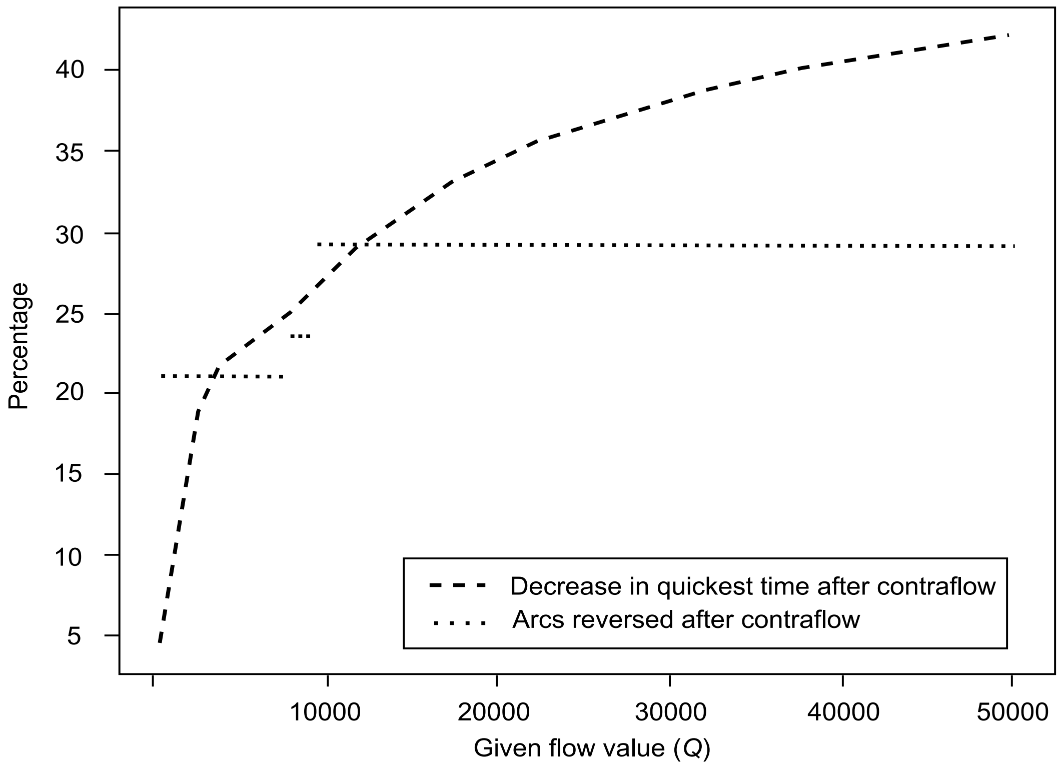


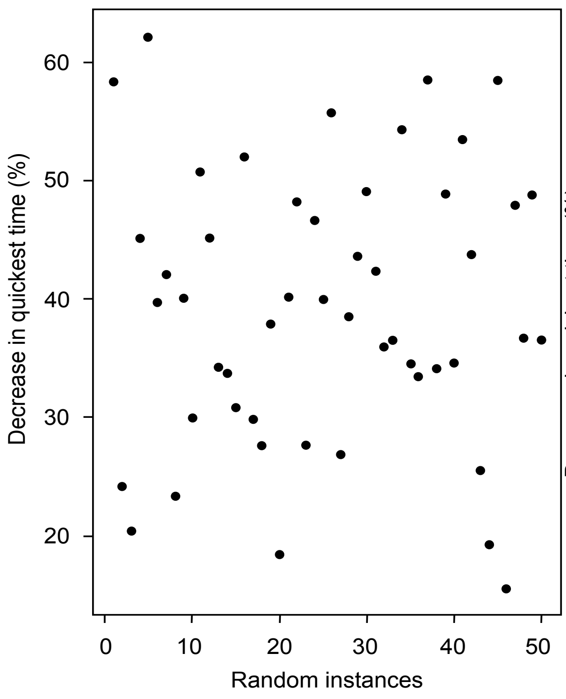
Figure 8: Percentage of arcs reversed and decrease in quickest Time

the percentage of decrease in the quickest time is found 42% as the number of cars reach 50000 (Figure 8).

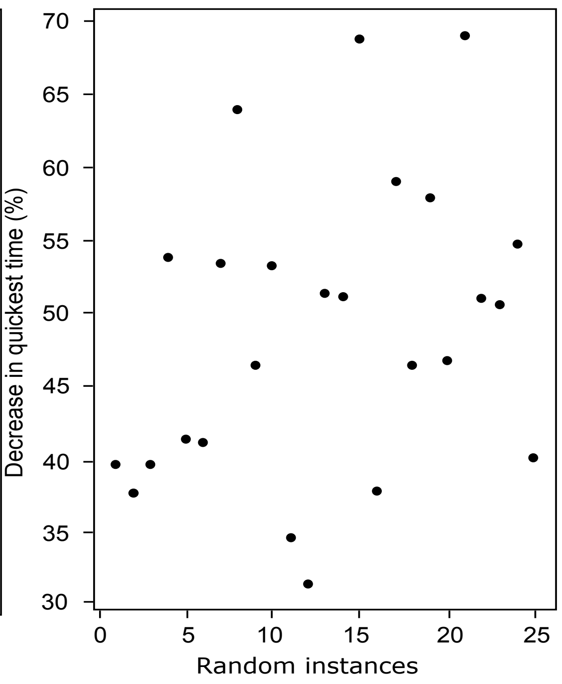
Next we randomly choose the number of evacuee-cars between 1 to 50 (in Thousands), capacity of an arc between 1 to 3 cars per second and travel time associated with arc between 1 to 10 minutes, and run 50 such instances (Figure 9a). The quickest time is found decreased by up to 62% with an average of 39.24 and standard deviation of 11.49. In this case, the average decrease in quickest time is significantly less than 50% (t stat $= -6.62$, P-value(one-tailed) $= 0.000$)). The flow rate is found increased by as high as 250%. The arc reversals range from 7.89% to 34.21%.

Keeping the number of evacuee-cars fixed to 100000 and randomizing other parameters as before (Figure 9b), the quickest time is found decreased highest by 69% with only 13% link reversals. The average of decrease in the percentage of quickest time is found 48.84 with standard deviation 10.12, and the arc reversals ranging from 13.16% to 31.58%. The average decrease in percentage of quickest time in this case is not significantly less than 50% (t-stat $= -0.572$, P-value(one-tailed) $= 0.2863$).

For a case study, we consider the road network of Kathmandu city inside ring road, consisting of major-roads (Fig 10 (a)). The evacuation is to be done from Pashupati Nath region (source), where a large gathering of people takes place in various religious occasions, to Tribhuvan University region (sink), where there is a sufficient open space.



(a) $1000 \leq Q \leq 50000$



(b) $Q = 100000$

Figure 9: Decrease in quickest time

The considered network has 46 nodes and 132 arcs. For the auto-based evacuation planning, we assume the capacity of each link ranging from 2 cars per second to 4 cars per second according to the width of the link. The travel time between any two nodes is as provided by Google Maps data.

Some of the computational results are listed in Table 1.

Table 1: Quickest time and flow comparison of Kathmandu network (before and after contraflow configuration)

	Quickest time (minutes)		Quickest flow rate (per minute)	
No. of cars	Before	After	Before	After
1,000	33.33	29.17	120	240
10,000	57.33	46.56	480	720
20,000	78.17	57.33	480	960
30,000	99.00	67.75	480	960
40,000	119.83	78.17	480	960
50,000	140.67	88.58	480	960

Although the routes of the quickest flow depend on the value of Q, they remain fixed after Q attains a sufficiently large value. In the above-mentioned case study, the routes of the quickest flow after contraflow configuration for $Q \geq 12000$ are given in Table 2.

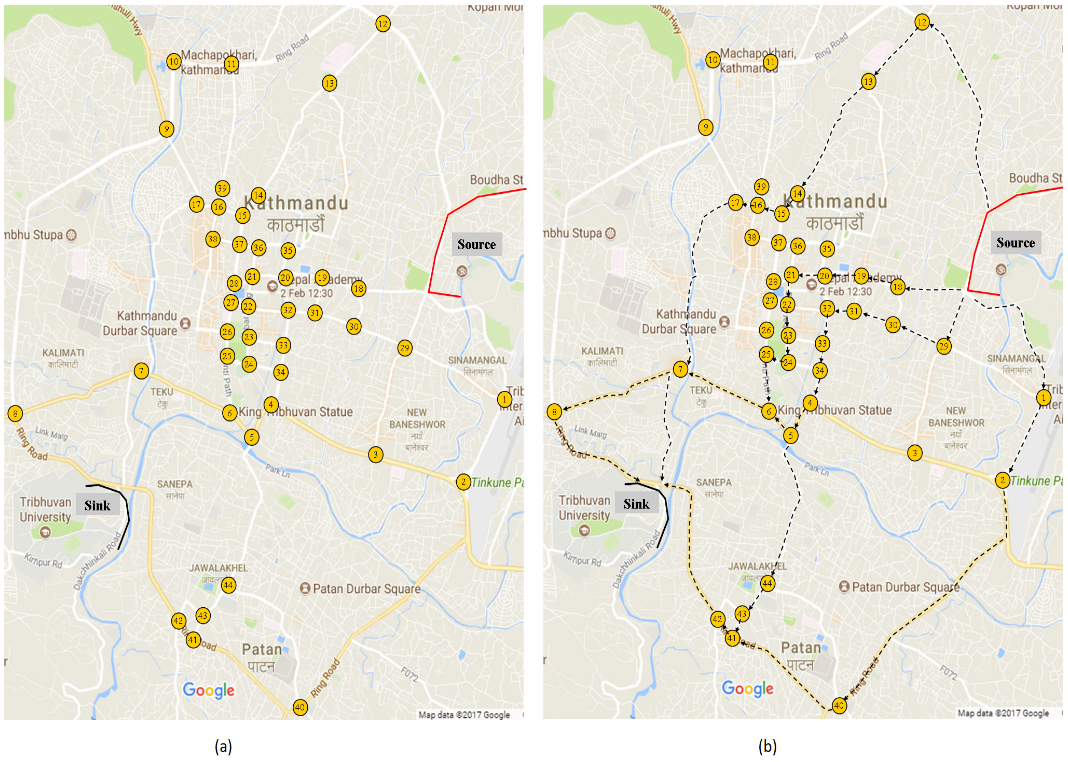


Figure 10: (a) Kathmandu network (b) Quickest contraflow solution

Table 2: Routes of the quickest flow solution for Kathmandu network ($Q \geq 12000$)

S.N.	Routes
1	Source – Narayan Gopal Chowk – Lainchour – Sorhakhutte – – Bishnumati Track – Kalimati – Kalanki – Sink
2	Route 1 up to Kalimati – University Path – Sink
3	Source – Gaushala – Kamal Pokhari – Teendhara Marga – Durbar Marga– –Bhadrakali – Sahid Gate – Tripureshwar – Kalimati – Kalanki – Sink
4	Route 3 upto Kalimati – University Path – Sink
5	Source – Gaushala – Purano Baneshwar – Putalisadak – Thapathali – – Kupondol – Jawalakhel – Ekantakuna – Dhobighat – Sink
6	Route 5 upto Thapathali – Tripureshwar – Kalimati – Kalanki – Sink
7	Route 6 upto Kalimati – University Path – Sink
8	Source – Koteshwar – Satdobato – Sink

As expected, because of the contraflow reconfiguration, the evacuation time decreases, and flow value increases significantly as the number of evacuees, and the network capacity increases. Some instances show that solutions without contraflow behave poorly if the lane direction do have a large capacity towards the source. We also notice that reversing complete arcs towards the sink unknowingly does not improve the solutions after a certain percentage of their reversals. Meaning that there should be instance dependent bound of the number of arcs to be reversed.

The coding of the above-mentioned computations is done in Python 3.6 and run in 64-bit Windows 10 operating system with Intel® Core™ i5 processor and 4GB RAM. The running time taken by our tests varies from few seconds for small-scale problems to a few minutes for large-scale problems (average running time on the Kathmandu network considered here is 98.5 seconds). Therefore, they are applicable in response or planning phases of the evacuation planning.

3.3 The Partial Contraflow Approach

In a transportation network, the reversal of an arc directs the traffic flow on the road-segment opposite to the pre-assigned direction. Since the capacity of an arc is proportional to the width of the road segment it represents, the portion of the road segment corresponding to the remaining capacity of the arc remains unoccupied by the flow and can be used for other purposes, e.g. to place the facilities to help evacuation. We denote the remaining capacity or the unused capacity of an arc (i, j) by r_{ij}. The problem of reversing a road segment up to the necessary capacity is termed as the partial contraflow problem (Pyakurel, Nath, Dempe, & Dhamala, 2019).

3.3.1 Static partial contraflow

In this section, we consider the partial contraflow problems corresponding to the static flow problems. Corresponding to the maximum static contraflow problem, we introduce the maximum static partial contraflow (MSPCF) problem (Definition 3.5) and corresponding the lex-maximum static flow problem, we introduce the lex-maximum partial contraflow problem (LMSPCF) (Definition 3.7). Thereafter, polynomial time algorithms are presented to solve these problems.

Definition 3.5 (Maximum static partial contraflow)**.** Given $N = (V, E, u, \tau, S, T)$, the maximum static flow by reversing the necessary arcs in E partially, recording the unused capacities, is the maximum static partial contraflow (MSPCF).

To solve MSPCF problem, we solve the maximum static flow problem in the auxiliary network, remove the positive flows in cycles, if any, reverse the road-segments up to the necessary capacities, and record the capacities of the segments not used by the flow. The procedure for a single-source-single-sink network is presented in Algorithm 2.

We prove the correctness of the algorithm in Theorem 3.4 and then discuss the complexity of MSPCF problem.

Theorem 3.4. *In a single-source-single-sink network, Algorithm 2 computes the*

```
Algorithm 2: MSPCF algorithm
Input  : Evacuation network $N = (V, E, u, \tau, s, t)$
Output: Maximum static partial contraflow in $N$
1 Construct the auxiliary network $N' = (V, E', u', \tau', s, t)$.
2 Find the maximum static flow $x$ in $N'$.
3 Decompose $x$ into chain and cycle flows. Update $x$ by removing cycle flows.
4 Reverse $(j, i) \in E$ proportional to the capacity $x_{ij} - u_{ij}$ iff $x_{ij} > u_{ij}$, $u_{ij}$ replaced
    by 0 whenever $(i, j) \notin E$. In reconfigured $N$, maximum static flow $= x$.
5 For each $(i, j) \in E$, if $(i, j)$ is reversed, then $r_{ij} = u'_{ij} - x_{ji}$ and $r_{ji} = 0$. If neither
    $(i, j)$ nor $(j, i)$ is reversed, $r_{ij} = u_{ij} - x_{ij}$.
```

maximum static partial contraflow and records the unused capacities correctly.

Proof. First we show that the flow x computed by the algorithm is feasible in N after reconfiguration. Then we show that x is optimal and the unused capacities are recorded correctly. Because of the removal of cycle flows in Step 3, for each $(i, j) \in E$, either $x_{ij} = 0$ or $x_{ji} = 0$. If $x_{ij} > u_{ij}$ then (j, i) is reversed proportional to the capacity $x_{ij} - u_{ij}$ in Step 4, and hence x becomes feasible in the reconfigured network. It is clear that x, in Step 2 is optimal in N'. Because of the conservation of flows at the intermediate nodes, the removal of cycle flows does not change the value of the flow updated in Step 3. Hence, x is optimal in N also after reconfiguration.

For the arcs $(i, j), (j, i)$ between nodes i and j, it is evident that either only one of them is reversed or both the them are not reversed. If (i, j) is reversed, it clearly indicates that $x_{ji} > u_{ji}$ there is no capacity of (j, i) unused, i.e. $r_{ji} = 0$, and

$$\begin{aligned} r_{ij} &= u_{ij} - (x_{ji} - u_{ji}) \\ &= u_{ij} + u_{ji} - x_{ji} \\ &= u'_{ij} - x_{ji}. \end{aligned}$$

If both are not reversed, then $x_{ij} \leq u_{ij}$ meaning $r_{ij} = u_{ij} - x_{ij}$. □

Steps 1, 4 and 5 of Algorithm 2 can be performed in $O(m)$ time. Step 3 takes $O(mn)$ time.

The time complexity of Step 2 is the time complexity of a maximum static flow algorithm. Maximum static flow algorithms have a long history but are still under study. The algorithms can be divided into two groups – augmenting path algorithms, and pre-flow push algorithms. An augmenting path with respect to a static flow x is a directed path from the source s to the sink t in the residual network N^x. A static flow x is maximum if and only if there does not exist an augmenting s–t path. Sending the flow along a flow augmenting path is referred to as flow augmentation. Following the labeling algorithm which runs in pseudo-polynomial time $O(mnU)$, where $U = \max\{u_{ij} : (i,j) \in E\}$, by Ford and Fulkerson (1962), there exist several improvements in the augmenting path algorithms. For example, the capacity scaling algorithm augments flows along paths with sufficiently large residual capacity and runs in $O(mn \log U)$ time; the shortest augmenting path algorithm, which augments the flows along shortest augmenting paths, runs in $O(mn^2)$ time (Ahuja & Orlin, 1991).

Preflow-push algorithms relax the flow conservation constraints at the intermediate steps, seek out the shortest paths, but send flows on individual arcs from active nodes (nodes with positive excess flow) rather than on the s–t paths. For each node, it maintains a distance label which, originally, is the shortest distance of the node from the sink. The label of the source node is maintained at n. A preflow-push algorithm selects an active node i with a label l, and either pushes the flow to a node with label $l-1$, or relabels i by $l+1$. The first preflow-push algorithm with running time $O(n^3)$ is due to Karzanov (1974). A generic pre-flow push algorithm runs in $O(mn^2)$ time (Goldberg & Tarjan, 1988). Among the several specific implementations of the algorithm, FIFO (first in first out) preflow-push algorithm Goldberg (1985) examines the active nodes in the FIFO order runs in $O(n^3)$ time, highest-label perflow push algorithm (Goldberg & Tarjan, 1988), which examines the active nodes with the highest distance label, runs in $O(n^2\sqrt{m})$ time, and the excess scaling algorithm (Ahuja & Orlin, 1989), with running time $O(mn + n^2 \log U)$, performs push/relabel operations at nodes with sufficiently large excesses and, among these nodes, selects node with the smallest distance label.

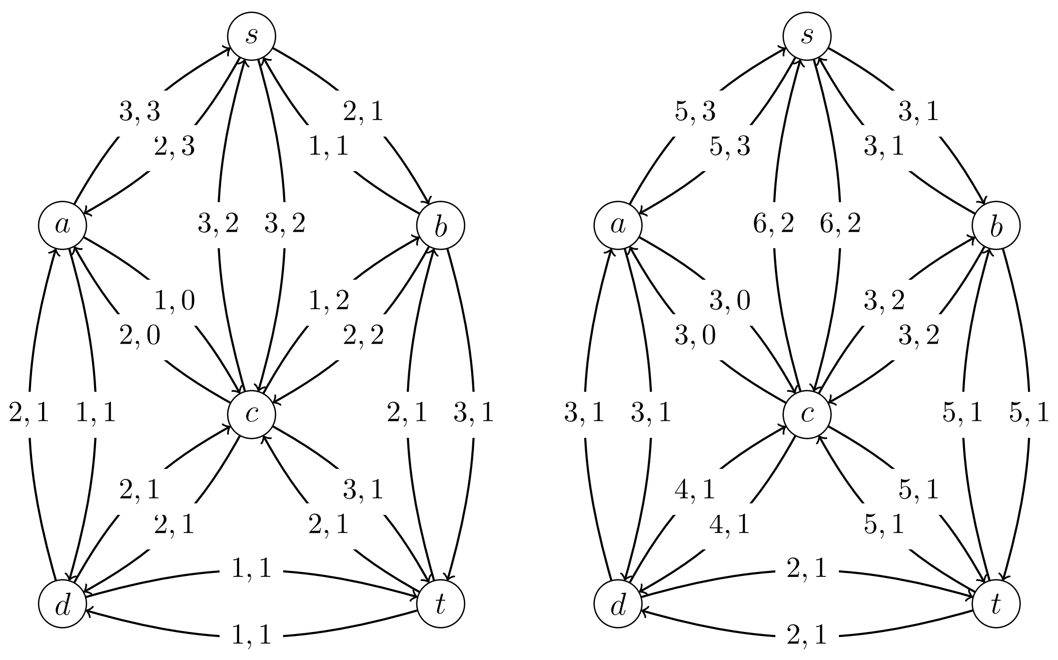
Among the various improvements in the algorithms to solve maximum static flow problem, the algorithm by Orlin (2013), solves the maximum flow problem in $O(mn)$ time if $m < n^{1.06}$.

In this way, we see that the running time of Algorithm 2 is dominated by the running times of Step 2 and Step 3. The maximum static flow in a multi-source-multi-sink network $N = (V, E, u, \tau, S, T)$ can be computed by adding two nodes s^*, t^* to N such that $u_{s^*s} = \infty, \forall s \in S$ and $u_{tt^*} = \infty, \forall t \in T$ and using the single-source-single-sink algorithm considering s^* as the source and t^* as the sink. The complexity of this computation remains the same in big O notation. Incorporating this idea in the MSPCF algorithm, we have the following result.

Theorem 3.5. *The MSPCF problem can be solved in strongly polynomial time.*

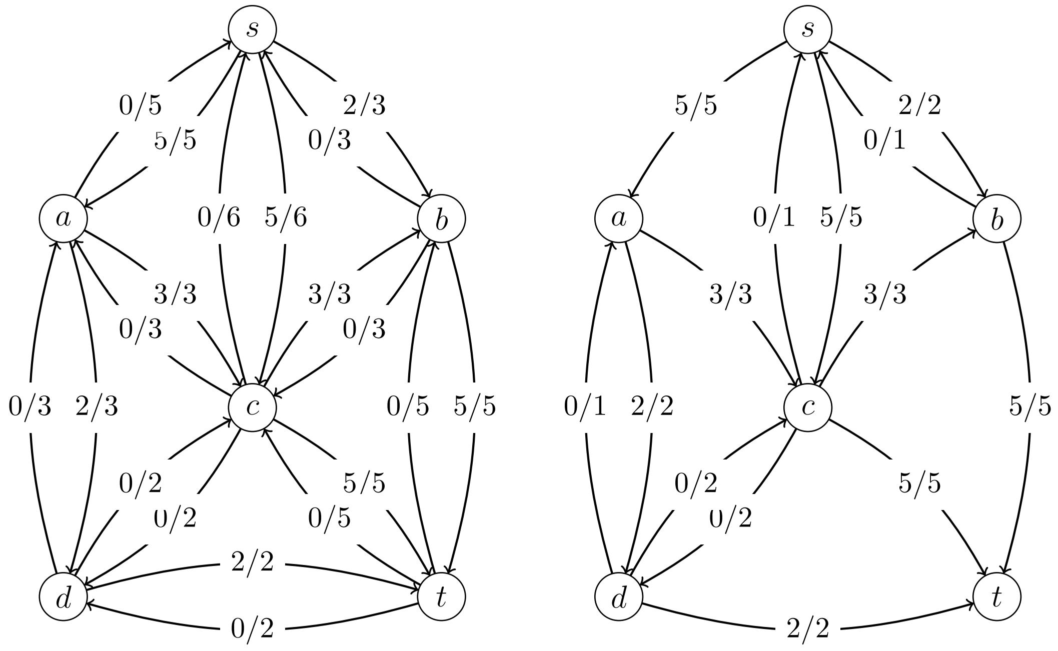
Now we present an example to illustrate the working of Algorithm 2.

Example 3.1. Consider an evacuation network N in Figure 11(a) in which s is the source and t is the sink. The arc labels represent capacity and traversal time. Although



(a) Network N **(b)** Auxiliary network N'

Figure 11: Evacuation network in Example 3.1



(a) Maximum static flow in N' **(b)** Max static flow in N after reconfigration

Figure 12: MSPCF solution

arc traversal time is not required for the static flow problems, we consider them to use the network in examples where the time is required. The maximum static flow value in this network is 7 (2 via path s–b–t, 3 via s–c–t, 1 via s–a–d–t, and 1 via s–a–c–b–t). Figure 11(b) represents the auxiliary network N' of N. The maximum static flow computation is shown in Figure 12(a). It can be seen that the value of the maximum static flow in N' is 12 (2 via path s–b–t, 5 via s–c–t, 2 via s–a–d–t, and 3 via s–a–c–b–t). The network N after partial contraflow configuration is shown in Figure 12(b). Arcs $(a, s), (c, a), (b, c), (t, c), (t, d)$ are reversed completely. The arc (c, s) is partially reversed upto the capacity 2, and the arc (d, a) is partially reversed upto the capacity 1. The unused capacities are: $r_{bs} = r_{cs} = r_{da} = 1, r_{cd} = r_{dc} = 2$.

In a multi-source-multi-sink network $N = (V, E, u, \tau, S, T)$, let $S_0 \subseteq S, T_0 \subseteq T$. For a given static flow x in N, we denote the maximum amount of flow that leaves S_0 by $v^{S_0}(x)$ and the maximum amount of flow that enters T_0 is denoted by $v^{T_0}(x)$.

Definition 3.6 (Lexicographic maximum static flow)**.** Consider an evacuation network $N = (V, E, u, \tau, S, T)$ with multiple sources or multiple sinks, such that $S \cap T = \varnothing$. Let $T_1 \subseteq \cdots, T_p \subseteq T$, then a maximum flow x that delivers a flow of value $v^{T_i}(x)$ into each $T_i (i = 1, \cdots p)$ is called a lexicographic maximum flow on the sinks. One can define a lexicographic maximum flow on the sources in a similar fashion (Minieka, 1973).

Construction of a lexicographic maximum static flow: Minieka (1973) proved the existence of the lexicographic maximum flows constructing such a flow iteratively in a network. To construct the flow on sinks, for the requirement of T_k, two extra nodes s^* and t^*, arcs $(i, s^*), (s^*, j)$ for each $i \in T \setminus (T_1 \cup \cdots \cup T_k), j \in S$, and arcs (l, t^*) for each $l \in T_1 \cup \cdots T_k$ are added. The newly added arcs are assigned infinite capacity. Initiating with the flow constructed in the previous iteration, Ford and Fulkerson algorithm is applied taking s^* as the source and t^* as the sink unless the flow is no more improved. Similar idea can be used to construct such a flow on sources. Details can be found in Minieka (1973).

Definition 3.7 (Lexicographic maximum static partial contraflow)**.** Given a multi-source-multi-sink network $N = (V, E, u, \tau, S, T)$, a lexicographic maximum static partial contraflow (LMSPCF) is the lexicographic maximum flow (on the sinks or on the sources) reversing the necessary arcs in E partially, recording the unused capacities.

To solve the LMSPCF problem, we present Algorithm 3. The idea is similar to that of Algorithm 2.

Analogous to Theorem 3.4, we can prove:

```
Algorithm 3: LMSPCF algorithm
Input  : Evacuation network $N = (V, E, u, \tau, S, T), S \cap T = \varnothing$
Output: Lexicographic maximum static partial contraflow in $N$
1 Construct the auxiliary network $N' = (V, E', u', \tau', S, T)$.
2 Find the lexicographic maximum static flow $x$ (on sources or sinks) in $N'$.
3 Decompose $x$ into chain and cycle flows. Update $x$ by removing cycle flows.
4 Reverse $(j, i) \in E$ proportional to the capacity $x_{ij} - u_{ij}$ iff $x_{ij} > u_{ij}$, $u_{ij}$ replaced
    by 0 whenever $(i, j) \notin E$. $x =$ lexicographic maximum static flow in reconfigured
    $N$.
5 For each $(i, j) \in E$, if $(i, j)$ is reversed, then $r_{ij} = u'_{ij} - x_{ji}$ and $r_{ji} = 0$. If neither
    $(i, j)$ nor $(j, i)$ is reversed, $r_{ij} = u_{ij} - x_{ij}$.
```

Theorem 3.6. *Algorithm 3 solves the LMSPCF correctly.*

Theorem 3.7. *The LMSPCF problem can be solved in a strongly polynomial time.*

Proof. As in Theorem 3.5, Steps 1, 4 and 5 of Algorithm 3 can be performed in $O(m)$ time, and Step 3 takes $O(mn)$ time. The lexicographic maximum flow in Step 2 can be performed using the flow construction suggested in Minieka (1973) discussed before Definition 3.7 in which Ford Fulkerson's algorithm is called in each iteration. The number of iterations is bounded by the number of sinks (sources) for the lexicographic maximal flows on sinks (sources). Using a strongly polynomial time algorithm (see the discussion before Theorem 3.5) instead of Ford Fulkerson's algorithm in each iteration, we can achieve a strongly polynomial time algorithm for LMSPCF. □

Example 3.2. Consider an evacuation network, as shown in Figure 13 with a source set $S = \{s_1, s_2\}$, and a sink set $T = \{t_1, t_2, t_3\}$. If we prioritize the sinks as t_1, t_2, t_3, i.e. $T_1 = \{t_1\}, T_2 = \{t_1, t_2\}, T_3 = T$, then an optimal solution with a partial contraflow configuration is shown in Figure 14(a). The maximum flow value at t_1 is 4, that at t_2 is 3, and at t_3, 7. Arcs $(t_1, a), (t_2, b), (b, s_2)$ are reversed fully, and each of $(b, a), (t_3, b)$ is reversed upto the capacity 1. The corresponding solution for the prioritization t_1, t_3, t_2, i.e. $T_1 = \{t_1\}, T_2 = \{t_1, t_3\}, T_3 = T$ is shown in Figure 14(b). In this case, the maximum flow values are 4 at t_1, 8 at t_3, and 2 at t_2, arcs $(t_1, a), (t_3, b), (b, s_2)$ being reversed fully, and each of $(b, a), (t_2, b)$ being reversed partially up to the capacity 1.

3.3.2 Dynamic partial contraflow

If we incorporate partial contraflow approach in the dynamic flow (flow over time) problems, we refer to the resulting problems as dynamic contraflow problems. We begin with the maximum dynamic contraflow problem.

Definition 3.8 (Maximum dynamic partial contraflow)**.** Given an evacuation network

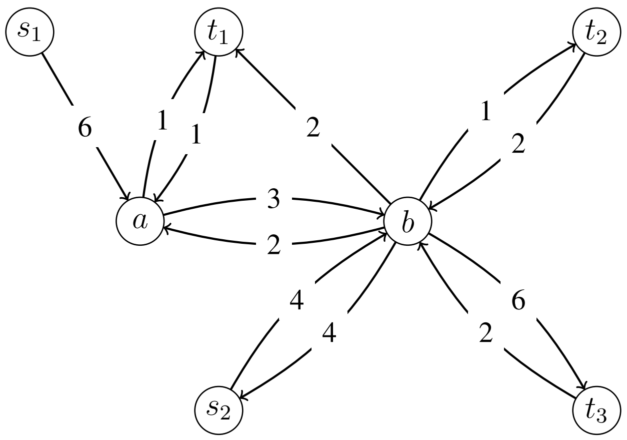


Figure 13: A network with source set $S = \{s_1, s_2\}$, sink set $T = \{t_1, t_2, t_3\}$

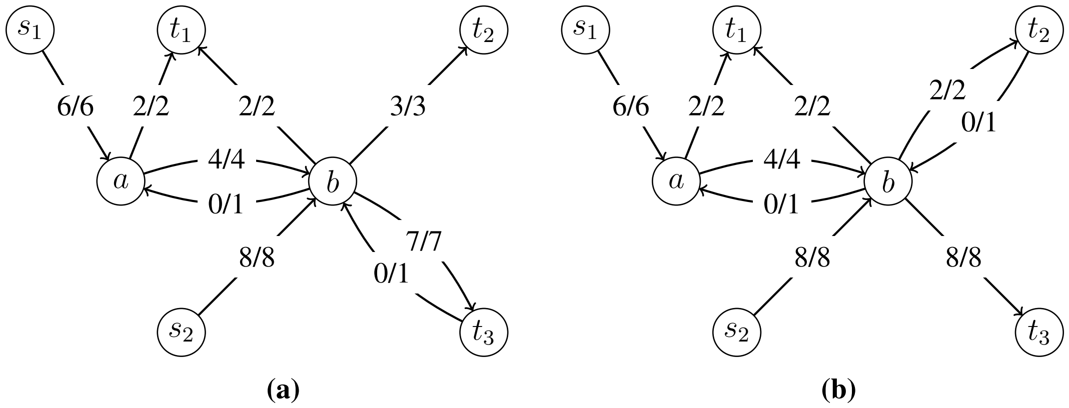


Figure 14: LMSPCF solution

$N = (V, E, u, \tau, s, t)$ with a time horizon θ, the maximum dynamic partial contraflow (MDPCF) is the maximum dynamic flow reversing the necessary arcs in E partially, at time zero, recording the capacities of the arcs not used by the flow.

To solve MDPCF, we solve the maximum dynamic flow problem (see problem (37), page 102) to identify static flow which gives a temporally repeated dynamic flow, and then reverse the arc whose reverse arc carries more flow than its capacity, as in the case of static partial contraflow problems. The procedure is given in Algorithm 4.

Algorithm 4: MDPCF algorithm

```
Input  : Evacuation network $N = (V, E, u, \tau, s, t)$, with a time horizon $\theta$
Output: Maximum dynamic partial contraflow in $N$
1 Construct the auxiliary network $N' = (V, E', u', \tau', s, t)$.
2 Find a static flow $x$ in $N'$ corresponding to the temporally repeated maximum
    dynamic flow with time horizon $\theta$.
3 Decompose $x$ into chain and cycle flows. Update $x$ by removing cycle flows.
4 Reverse $(j, i) \in E$ proportional to the capacity $x_{ij} - u_{ij}$ iff $x_{ij} > u_{ij}$, $u_{ij}$ replaced
    by 0 whenever $(i, j) \notin E$. In reconfigured $N$, maximum dynamic flow =
    temporally repeated dynamic flow induced by $x$ with the time horizon $\theta$.
5 For each $(i, j) \in E$, if $(i, j)$ is reversed, then $r_{ij} = u'_{ij} - x_{ji}$ and $r_{ji} = 0$. If neither
    $(i, j)$ nor $(j, i)$ is reversed, $r_{ij} = u_{ij} - x_{ij}$.
```

The feasibility of Algorithm 4 can be established by arguments analogous to those given in the proof of Theorem 3.2. The crucial step of this algorithm is Step 2 which requires to calculate the static flow, the temporal repetition of which gives a maximum dynamic flow in the auxiliary network N'. For a time horizon θ, this can be done by finding the static flow x that maximizes $\theta v(x) - \sum_{(i,j)\in E} \tau_{ij} x_{ij}$ which is equivalent to minimizing $-\theta v(x) + \sum_{(i,j)\in E} \tau_{ij} x_{ij}$ (Ford & Fulkerson, 1958). Hence, x in Step 2 can be obtained by adding an extra arc (t, s) with $\tau'_{ts} = -\theta, u'_{ts} = \infty$ to N' and solving a minimum cost circulation problem taking τ' as the cost. A minimum cost circulation is a special type of minimum cost flow.

In a network N with supplies and demands associated with its nodes, for any real number π_i associated with a node i in N_x, a reduced cost associated with each (i, j) in N_x is defined as

$$c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$$

where c_{ij} is the cost associated with the arc (i, j). We write $C = \max\{c_{ij} : (i, j) \in E\}$ as mentioned earlier also. A solution x of a minimum cost flow problem, is optimal if and only if the following equivalent conditions hold.

- Negative cycle optimality conditions: N^x does not contain a negative cost directed cycle.

- Reduced cost optimality conditions: $c^{\pi}_{ij} \geq 0$ for all $(i,j) \in N_x$.
- Complementary slackness optimality conditions:

  $c^{\pi}_{ij} > 0 \Rightarrow x_{ij} = 0$,

  $0 < x_{ij} < u_{ij} \Rightarrow c^{\pi}_{ij} = 0$, and

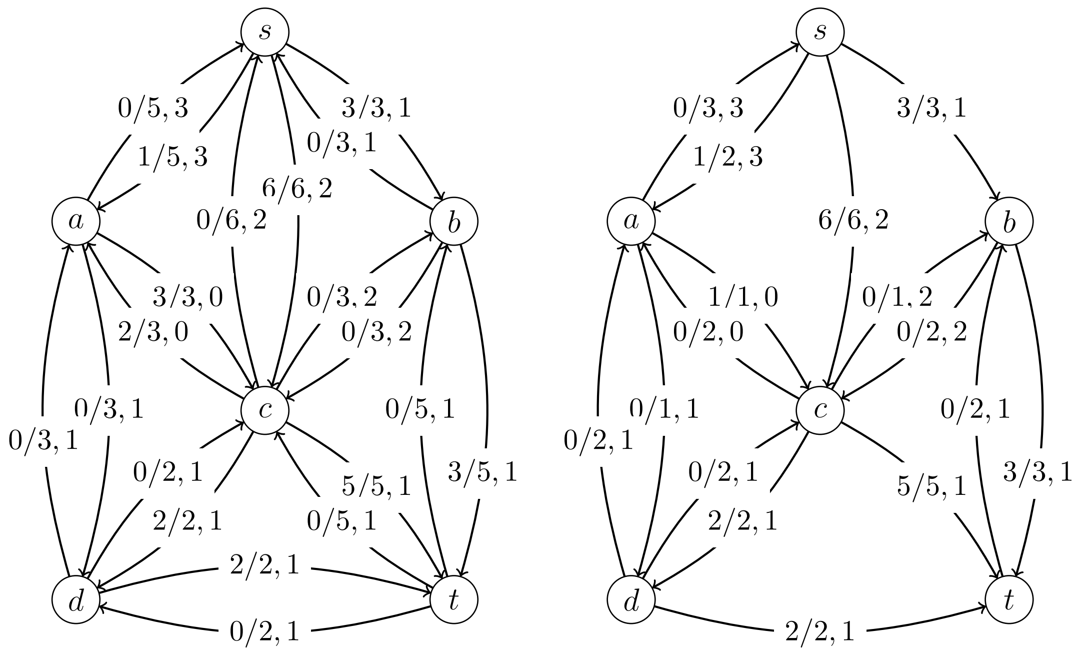
  $c^{\pi}_{ij} < 0 \Rightarrow x_{ij} = u_{ij}$.

The algorithms of solving the minimum cost flow problem are based on these optimality conditions.

Since a minimum cost flow problem is a linear programming problem, the simplex method can be used to solve it but a general simplex method may not be practically efficient. Network simplex algorithm (Dantzig, 1998), a highly efficient algorithm in practice, exploits the special structure of the problem interpreting the core concepts of the simplex method as network flow operations.

The primal-dual algorithm (Ford & Fulkerson, 1962) maintains a pseudoflow satisfying the optimality conditions and attempts to reduce primal infeasibility by the maximum amount solving a maximum flow problem. The successive shortest path algorithm (Jewell, 1958) maintains a pseudoflow satisfying the optimality conditions and augments the flow along the shortest paths from excess nodes to deficit nodes in the residual network corresponding to the pseudoflow, runs in pseudopolinomial time. However using capacity scaling (Edmonds & Karp, 1972), one can achieve polynomial time running time. A version of this algorithm by Orlin (1993) runs in $O(m \log U(m + n \log n))$ time. The enhanced capacity scaling algorithm by Orlin (1993) runs in $O(m \log n(m + n \log n))$ time. Augmenting the flow along negative length directed cycles in the residual network, the cycle canceling algorithm by Klein (1967) achieves the optimal flow as soon as there remains no negative cost cycle. Although, it runs in pseudopolynomial time, among the improved versions of it, the minimum-mean cycle canceling algorithm by Goldberg and Tarjan (1989) runs in a strongly polynomial $O(n^2 m^3 \log n)$ time, and using the dynamic tree data structure, their cancel-and-tighten algorithm runs in $O(nm \log n \min\{\log(nC), m \log n\})$ time for integer arc-costs, and in $O(nm^2 \log^2 n)$ time for arbitrary real-valued arc costs. An adaption of the pre-flow push algorithm of maximum flow, the implementation by Goldberg and Tarjan (1987) of the cost scaling algorithm originally given by Röck (1980) uses $O(\log(nC))$ max flow operations.

The algorithms mentioned above are only a few of the minimum cost flow algorithms available in literature. Although the problem is well known and well solved, research in the area is still active, the current work by Hu, Zhao, Liu, Liang, and Ma (2020) is an example.



(a) Optimal static flow in N' **(b)** Optimal static flow in N after reconfigration

Figure 15: MDPCF solution, $\theta = 6$

Use of any of the available strongly polynomial time algorithm in Step 2 makes the running time of Algorithm 4 strongly polynomial and we can safely say that:

Theorem 3.8. *The MDPCF problem can be solved in strongly polynomial time.*

Example 3.3. Consider the network in Figure 11(a) again. For $\theta = 6$, the value of the maximum dynamic flow is 18, described as follows.

Path (P)	$\tau(P)$	x^P	$\theta - \tau(P)$	Dynamic flow value
s–b–t	2	2	4	8
s–c–t	3	3	3	9
s–a–c–d–t	5	1	1	1

To find the solution to the MDPCF problem, first we add the arc (t, s) with $u_{ts} = \infty, \tau_{ts} = -\theta$ to N' and find the minimum cost circulation on it. In this way, we get a static flow in N' corresponding to the temporally repeated dynamic flow as shown in Figure 15(a). We remove the cycle flow in the cycle a–c–a, and the resulting flow in N after reconfiguration is shown in Figure 15(b). The value of the maximum dynamic flow after reconfiguration is 30 as described in the following table.

Path (P)	$\tau(P)$	x^P	$\theta - \tau(P)$	Dynamic flow value
s–b–t	2	3	4	12
s–c–t	3	5	3	15
s–c–d–t	4	1	2	2
s–a–c–d–t	5	1	1	1

The arcs $(b,s),(c,s),(t,c),(t,d)$ are to be reversed to the full capacity.

Definition 3.9 (Quickest partial contraflow)**.** Given $N=(V,E,u,\tau,s,t)$ and a supply Q at s, the quickest partial contraflow (QPCF) is the quickest flow allowing the arcs to be reversed partially, at time zero, and recording unused arc capacities.

We can use the procedure in Algorithm 1 to solve QPCF replacing Step 4 by that of Algorithm 4 and adding Step 5 of Algorithm 4 at last. The correctness of the algorithm can be justified similarly. As in the proof of Theorem 3.3, we can show that the QPCF problem can be solved in $O(nm^2 \log^2 n)$ time. Hence, we have:

Theorem 3.9. *There exists a strongly polynomial time algorithm to solve the QPCF.*

3.4 Orientation Dependent Transit Times

So far we have considered situations in which the time of travel on an arc remains same after its reversal. There are situations when the time of travel varies when the direction of the traffic flow is reversed in road networks depending on the topography, etc. To capture such a situation, we consider travel time functions $\tau, \overleftarrow{\tau} : E \to \mathbb{R}_{\geq 0}$ For each $e \in E$, with tail i and head j, $\tau(e)$ denotes the arc transit time from i to j and $\overleftarrow{\tau}(e)$ denotes transit time from j to i.

Without loss of generality, we make the following conventions.

1. Whenever $(i,j) \in E$, there is $(j,i) \in E$. This can be done, for our purpose, by assigning $u_{ji} = 0$ if such an arc does not exist.
2. Defining $\tau(j,i) = \overleftarrow{\tau}(i,j)$, we assume the existence of only one travel time function τ depending on the orientation of arcs.

Remark 3.1. In case the above-mentioned conventions are not satisfied, we can use suitable network transformations to meet the requirements.

Example 3.4. Consider a network in Figure 16 (i). The arc labels represent the capacity of the arc and transit time on the direction of the arc. For a time horizon of 7 units, a maximum of 8 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, 1 along s–a–b–d thrice) can be sent from s to d without allowing arc reversals.

If (b,a) is reversed and the transit time is kept intact (Figure 16 (ii)), the maximum of 10 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, 1 along s–a–b–d (that contains original (a,b) thrice), and 2 along s–a–b–d (that contains (b,a) reversed) once).

If the transit time depends on the orientation, when (b,a) is reversed its transit time becomes that of original (a,b) (Figure 16 (iii)). In this case the maximum of 14 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, 3 along s–a–b–d thrice). In this case,

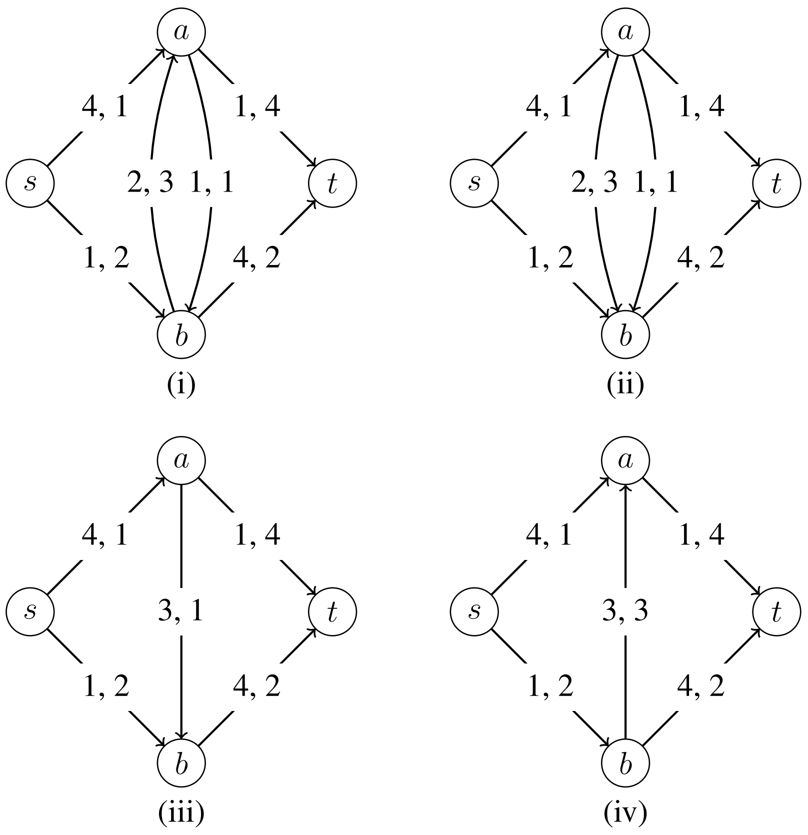


Figure 16: Different cases of arc reversals

we can add the capacities of $(a, b), (b, a)$ and replace the two arcs by a single arc (a, b). If (a, b) is reversed, however, the maximum flow value reduces to 5 (Figure 16(iv)).

To solve the problems of optimizing the flow with arc reversals in a network with orientation dependent transit times, we construct the auxiliary network, as described in the following definition.

Definition 3.10. Given $N = (V, E, u, \tau, s, t)$, with orientation dependent transit times, we define the auxiliary network as $N' = (V', E', u', \tau', s, t)$ in which

1. $V' = V, E' = E$
2. $\forall(i, j) \in A', u'(i, j) = u(i, j) + u(j, i)$
3. $\forall(i, j) \in A', \tau'(i, j) = \tau(i, j)$

Constructing the auxiliary network as given in Definition 3.10, the contraflow algorithms discussed above work well to solve the corresponding problems with orientation dependent transit times (Nath, Pyakurel, & Dhamala, 2021), and we have

Theorem 3.10. *The maximum dynamic contraflow problem and the quickest contraflow problem, with orientation dependent transit times, and the corresponding partial contraflow problems can be solved in strongly polynomial times.*

3.5 Contraflow in Abstract Networks

In Section 2.4, we have seen that flows in abstract networks generalize the flows in (classical) networks. The concept of contraflow in abstract networks has been introduced by Pyakurel et al. (2017). In this Section, we adapt the idea to design partial contraflow algorithms in abstract networks. The results are based on our publication Pyakurel, Nath, and Dhamala (2019).

To model an evacuation problem using abstract network, we represent a portion of the road between two intersections by an element pair $(\overrightarrow{e}, \overleftarrow{e})$ representing road segments with opposite directions of traffic flow. Let S, T be the set of terminals (sources and sinks) respectively. For $s \in S, t \in T$, we denote the collections of s–t paths by $\overrightarrow{\mathscr{P}}$ and that of t–s paths by $\overleftarrow{\mathscr{P}}$. We assume that for each $\overrightarrow{P} = \{s, e_1, e_2, \dots e_n, t\} \in \overrightarrow{\mathscr{P}}$, there is $\overleftarrow{P} = \{t, e'_n, e'_{n-1}, \dots e'_1, s\} \in \overleftarrow{\mathscr{P}}$, and vice versa, where if $e_i = \overrightarrow{e}$, then $e'_i = \overleftarrow{e}$ and if $e_i = \overleftarrow{e}$, then $e'_i = \overrightarrow{e}$. We write $\overleftrightarrow{\mathscr{P}} = \overrightarrow{\mathscr{P}} \cup \overleftarrow{\mathscr{P}}$ and represent the evacuation network by $\mathcal{N} = (\mathcal{E}, \overleftrightarrow{\mathscr{P}}, u, \tau, S, T)$ where u, τ are capacity, and travel time functions. We also assume that $\tau(\overleftarrow{e}) = \tau(\overrightarrow{e})$, $\tau(s) = \tau(t) = 0\ \forall s \in S, t \in T$, and that $(\mathcal{E}, \overrightarrow{\mathscr{P}})$ is an abstract network. In a single-source-single-sink network with $S = \{s\}, T = \{t\}$, we write the network as $\mathcal{N} = (\mathcal{E}, \overleftrightarrow{\mathscr{P}}, u, \tau, s, t)$.

To design the algorithms for contraflow, with a point of view that all t–s paths can be reversed, we make use of the auxiliary network $\overline{\mathcal{N}} = (\overline{\mathcal{E}}, \overline{\mathscr{P}}, \overline{u}, \overline{\tau}, S, T)$ of the evacuation network $\mathcal{N} = (\mathcal{E}, \overleftrightarrow{\mathscr{P}}, u, \tau, S, T)$ as follows. The element set $\overline{\mathcal{E}}$ consists of all the elements of S, T and each pair of elements $(\overrightarrow{e}, \overleftarrow{e})$ from $\mathcal{E}$ replaced by an element $\overline{e}$. We replace $\overrightarrow{e}$ and $\overleftarrow{e}$ in each path in $\overrightarrow{\mathscr{P}}$ by $\overline{e}$ to get a collection of s–t paths $\overline{\mathscr{P}}$. The travel time and capacity of $\overline{e}$ in the transformed network are, respectively,

$$\overline{\tau}(\overline{e}) = \overline{\tau}(\overrightarrow{e}) = \tau(\overleftarrow{e}) \text{ and } \overline{u}(\overline{e}) = u(\overrightarrow{e}) + u(\overleftarrow{e})$$

It is easy to check that $(\overline{\mathcal{E}}, \overline{\mathscr{P}})$ is an abstract network if $(\mathcal{E}, \overleftarrow{\mathscr{P}})$ is an abstract network.

Example 3.5. Let us consider an evacuation network as shown in Figure 11(a). Take the element set as

$$\mathcal{E} = \{s, t, sa, as, sb, bs, sc, cs, ac, ca, ad, da, bc, cb, bt, tb, cd, dc, ct, tc, dt, td\}$$

where the elements ij, ji represent the two-way road segment between i and j, s is the source element and t is the sink element. We consider the collection of s–t paths, explicitly, as

$$\overrightarrow{\mathscr{P}} = \{\overrightarrow{P_1}, \overrightarrow{P_2}, \overrightarrow{P_3}, \cdots \overrightarrow{P}_{14}\},$$

where

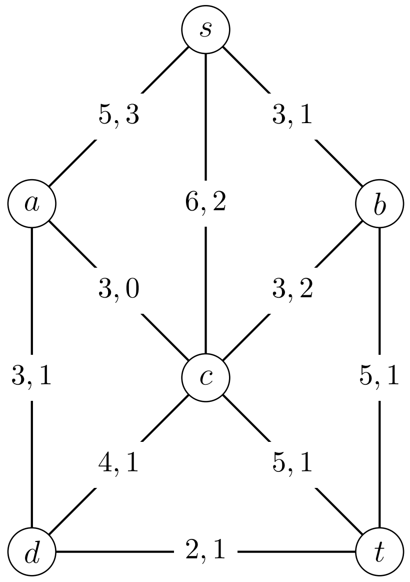


Figure 17: Auxiliary abstract network in Example 3.5

$\overrightarrow{P}_1 = (s, sb, bt, t)$, $\overrightarrow{P}_2 = (s, sc, ct, t)$, $\overrightarrow{P}_3 = (s, sb, bc, ct, t)$,
$\overrightarrow{P}_4 = (s, sa, ac, ct, t)$, $\overrightarrow{P}_5 = (s, sc, cb, bt, t)$, $\overrightarrow{P}_6 = (s, sa, ad, dt, t)$,
$\overrightarrow{P}_7 = (s, sc, cd, dt, t)$, $\overrightarrow{P}_8 = (s, sb, bc, cd, dt, t)$, $\overrightarrow{P}_9 = (s, sa, ac, cb, bt, t)$,
$\overrightarrow{P}_{10} = (s, sa, ad, dc, ct, t)$, $\overrightarrow{P}_{11} = (s, sc, ca, ad, dt, t)$, $\overrightarrow{P}_{12} = (s, sa, ac, cd, dt, t)$,
$\overrightarrow{P}_{13} = (s, sb, bc, ca, ad, dt, t)$, $\overrightarrow{P}_{14} = (s, sa, ad, dc, cb, bt, t)$.

In the auxiliary abstract network as shown in Fig. 17, the element set becomes

$$\mathcal{E} = \{s, t, \overline{sa}, \overline{sb}, \overline{sc}, \overline{ac}, \overline{ad}, \overline{bc}, \overline{bt}, \overline{cd}, \overline{ct}, \overline{dt}\}$$

and the set of paths $\overline{\mathscr{P}} = \{\overline{P}_1, \cdots, \overline{P}_{14}\}$ where $\overline{ij} \in \overline{P}_k$ is obtained by replacing ij and ji in $\overrightarrow{P}_k$. For example, $\overline{P}_{11} = \{s, \overline{sc}, \overline{ac}, \overline{ad}, \overline{dt}, t\}$, $\overline{P}_{12} = \{s, \overline{sa}, \overline{ac}, \overline{cd}, \overline{dt}, t\}$. We see that the paths in $\overline{\mathscr{P}}$ satisfy the switching property, e.g., $\overline{P}_{11}$ and $\overline{P}_{12}$ intersect at $\overline{ac}$ and $\overline{P}_7 \subseteq \overline{P}_{11} \times^{\overline{ac}} \overline{P}_{12}$, $\overline{P}_6 \subseteq \overline{P}_{12} \times^{\overline{ac}} \overline{P}_{11}$.

If $x : \mathscr{P} \to \mathbb{R}_{\geq 0}$ is a static flow in the abstract network $\mathcal{N} = (\mathcal{E}, \mathcal{P}, u, \tau, S, T)$, then we define $\chi : \mathcal{E} \to \mathbb{R}_{\geq 0}$ by

$$\chi(e) = \sum_{P \in \mathscr{P} : e \in P} x(P), \forall e \in \mathcal{E} \tag{19}$$

which is the amount of flow that is assigned to e. Now, we present Algorithm 5 to solve maximum static partial contraflow problem in abstract networks.

Applying the algorithm of Kappmeier (2015) to find the lexicographic maximum static flow in Step 2, we can easily convert Algorithm 5 to solve the lexicographic abstract maximum static contraflow problem if the terminals have orders with the restrictions mentioned in Kappmeier et al. (2014).

According to Kappmeier et al. (2014), in an abstract network $(\mathcal{E}, \mathscr{P})$, the maximum

Algorithm 5: Abstract maximum static partial contraflow algorithm

```
Input  : Evacuation network $\mathcal{N} = (\mathcal{E}, \overleftrightarrow{\mathscr{P}}, u, \tau, S, T)$ such that $(\mathcal{E}, \overleftarrow{\mathscr{P}})$ is an
         abstract network
Output: Abstract maximum static partial contraflow
1 Construct the auxiliary network $\overline{\mathcal{N}} = (\overline{\mathcal{E}}, \overline{\mathscr{P}}, \overline{u}, \overline{\tau}, S, T)$.
2 Find abstract maximum static flow $x$ in $\overline{\mathcal{N}}$ using the algorithm of McCormick
   (1996).
3 For each $\overline{e} \in \overline{\mathcal{E}}$, calculate the unused capacity $r(\overline{e}) = \overline{u} - \chi(\overline{e})$.
```

dynamic abstract flow problem with time horizon θ (in the temporally repeated form) can be solved by solving the following weighted abstract maximum flow problem

$$\max \quad \sum_{P \in \mathscr{P}} \left[\theta - \sum_{e \in P} \tau(e) \right] x(P) \tag{20}$$

subject to

$$\sum_{P \in \mathscr{P} : e \in P} x(P) \quad \leq \quad u(e), \forall e \in \mathcal{E} \tag{20a}$$

$$x(P) \quad \geq \quad 0, \forall P \in \mathscr{P} \tag{20b}$$

Problem (20) can be solved by using the algorithm given in Martens and McCormick (2008). Using their algorithm as a subroutine, we construct Algorithm 6.

Algorithm 6: Abstract maximum dynamic partial contraflow algorithm

```
Input  : Evacuation network $\mathcal{N} = (\mathcal{E}, \overleftrightarrow{\mathscr{P}}, u, \tau, S, T)$ such that $(\mathcal{E}, \overleftarrow{\mathscr{P}})$ is an
         abstract network, and a time horizon $\theta$
Output: Abstract maximum dynamic partial contraflow
1 Construct the auxiliary network $\overline{\mathcal{N}} = (\overline{\mathcal{E}}, \overline{\mathscr{P}}, \overline{u}, \overline{\tau}, S, T)$.
2 Solve the problem (20) in $\overline{\mathcal{N}}$ using the algorithm of Martens and McCormick
   (2008) to find the abstract static flow $x$.
3 For each $\overline{e} \in \overline{\mathcal{E}}$, calculate the unused capacity $r(\overline{e}) = \overline{u} - \chi(\overline{e})$.
4 The abstract dynamic partial contraflow $f$ can be constructed using
      $f(\overline{P}_\xi) = x(\overline{P}), 0 \leq \xi < \theta - \sum_{e \in P} \tau(e)$
```

Since the running times of Step 2 of Algorithm 5 and Algorithm 6 is polynomial, we conclude the following:

Theorem 3.11. *Abstract maximum static partial contraflow algorithm and abstract maximum dynamic partial contraflow algorithm run in polynomial times.*

CHAPTER 4

MODELS WITH VARIABLE TRANSIT TIMES

4.1 Introduction

In the models considered in the previous chapters, we have assumed that the transit time on an arc is constant. However, it is a common experience that the transit time on a road segment increases with the increase of congestion of the traffic in it. The constant time models do not take into account the current flow situation on the arc.

Attempts to capture the dependency of the travel time on the flow leads to flow-dependent models. A fully realistic model of flow-dependent transit times on arcs has to consider density, speed, and flow rate evolving along an arc to determine the transit time. According to Langkau (2003), the literature is devoid of algorithmic techniques to obtain optimal solutions, taking all the parameters into consideration, even for networks of modest size.

In this chapter, we study models in which the transit time depends either on inflow rate or the density of the flow and develop algorithms for partial contraflow configuration.

4.1.1 Transit time functions

The commonly used transit time functions, also known as link performance functions, showing the dependency of transit time on the arc-flow are BPR function developed by US Bureau of Public Roads, and Davidson's function (Sheffi, 1985).

In BPR function, the transit time for an arc (i, j) is given by

$$\tau_{ij} = \tau_{ij}^{0} \left[1 + \alpha \left(\frac{x_{ij}}{u_{ij}^{\text{pr}}} \right)^{\beta} \right] \tag{21}$$

where τ_{ij}^{0} is the free-flow transit time, u_{ij}^{pr} is the practical capacity of the arc. The parameter α is the ratio of the transit time per unit distance at practical capacity to that at free flow, and β is the parameter which is the measurement of how fast the estimated

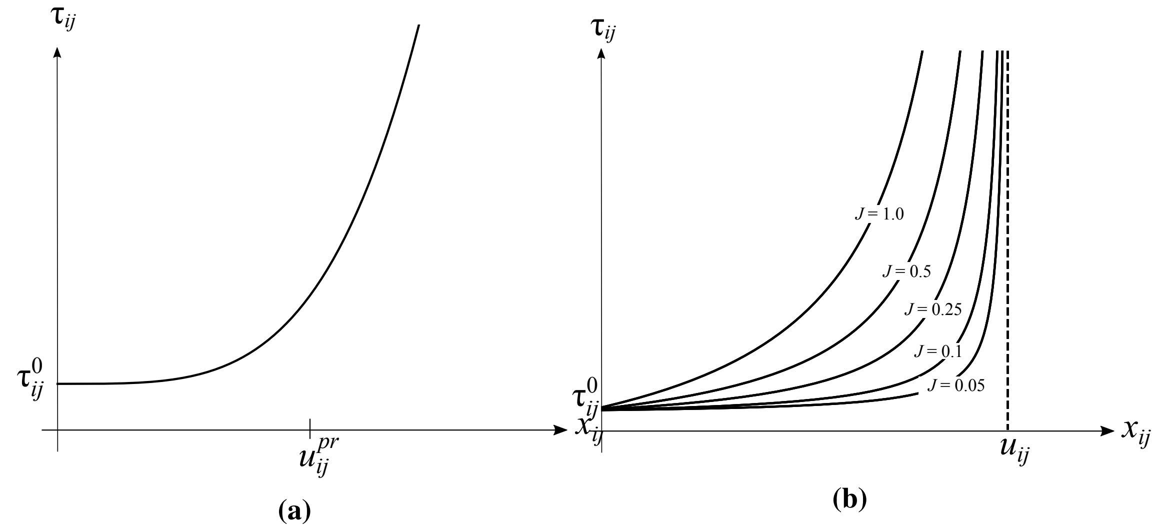


Figure 18: (a) BPR function (b) Davidson's function

average speed decreases from free flow to congested conditions. Generally, α, β are taken to be $0.15, 4.0$ respectively.

A BPR function is not asymptotic to the line $\tau_{ij} = u_{ij}$ (Figure 18(a)) which does not go with an assumption, considered in traffic flow theory, that the flow-dependent transit time increases infinitely when the flow rate reaches towards the capacity. Based on queueing theory, the Davidson's function defined as

$$\tau_{ij} = \tau^0_{ij} \left[1 + J \frac{x_{ij}}{u_{ij} - x_{ij}} \right] \tag{22}$$

is asymptotic to the line $\tau_{ij} = u_{ij}$ (Figure 18(b)). The parameter J, called a delay parameter, depends on the quality of the road (Mtoi & Moses, 2014). It determines the shape of the curve and can be calculated with the field measurements.

4.1.2 Dynamic flow problems with flow-dependent transit times

In the model described in Section 2.3, if the transit time τ, instead of being constant, depends in the flow, then such a model is referred to as a dynamic flow model or a flow over time model with flow-dependent transit time. The temporally repeated flow defined in case of constant transit times can be generalized to the flow-dependent cases.

Flow-dependent temporally repeated flow: Given a network $N = (V, E, u, \tau, s, t)$, let x be a feasible static flow, and τ depend on the flow. Suppose that $\mathcal{P}$ is the set of s–t paths such that the transit time on the path P, $\tau_P(x) \leq \theta, \forall P \in \mathcal{P}$. The temporally repeated dynamic flow X with flow-dependent transit times within the time horizon θ is defined as follows.

i) The transit time of each $(i,j) \in E$ is fixed to $\tau_{ij}(x_{ij})$ so that flow entering (i,j) at time ξ reaches j at time $\tau_{ij}(x_{ij}) + \xi, \forall \xi \in [0,\theta)$.

ii) The flow X enters $P \in \mathcal{P}$ at a constant rate x^P starting at time zero and ending at time $\theta - \tau_P(x)$.

The value of the temporally repeated flow defined above is

$$\sum_{P \in \mathcal{P}} (\theta - \tau^P(x))x^P = \theta v(x) - \sum_{(i,j)\in E} \tau_{ij}(x_{ij})x_{ij} \tag{23}$$

which means that, as in the case of fixed transit time case, the value of the temporally repeated flow with flow-dependent transit times is independent of the path decomposition.

In what follows, we discuss dynamic flow models with two types of flow-dependent transit times:

- Inflow-dependent transit times (IFDTT)
- Load-dependent transit times (LDTT)

4.2 Dynamic Flow with IFDTT

In the dynamic flow models with inflow-dependent transit times (IFDTT), the transit time of flow on an arc is determined when the flow enters the arc and depends on the inflow rate at that time. It is assumed that each arc $(i,j) \in E$ has an associated non-negative, non-decreasing, piece-wise constant, left-continuous transit time function $\tau_{ij} : [0, u_{ij}] \to \mathbb{R}_{\geq 0}$ which denotes the time taken by the flow to traverse arc (i,j). Any general transit time function can be approximated, within an arbitrary precision, by a function meeting aforementioned requirements.

4.2.1 The bow network

To solve some dynamic flow problems with inflow-dependent transit times, Langkau (2003) expand the given network to construct a bow network with the intuition that if a road segment has multiple lanes, a driver prefers to choose the fastest lane available. That means, the slower lane is chosen only after the immediate faster lane is full. Once a lane is chosen, the speed of the in vehicle is fixed to the upper capacity of the lane.

We denote the bow network of $N = (V, E, u, \tau)$ as $N^b = (V^b, E^b, u^b, \tau^b)$. For each

$e = (i, j) \in E, 0 = u^0 < u^1 < \cdots < u^k = u_{ij}$, let the transit time function be given as

$$\tau_{ij}(x) = \begin{cases} \tau^1, & u^0 < x \leq u^1 \\ \tau^2, & u^1 < x \leq u^2 \\ \cdots \\ \tau^k, & u^{k-1} < x \leq u^k. \end{cases}$$

Corresponding to each $e = (i, j)$, the node set V^b contains the nodes $i = e_0, e_1, \cdots, e_k, j$, and the arc set E^b contains arcs

$$(e_0, e_1), \cdots, (e_{k-1}, e_k), (e_k, j)$$

called regulating arcs with

$$u^b_{e_0e_1} = u^k, u^b_{e_1e_2} = u^{k-1}, \cdots, u^b_{e_{k-1}e_k} = u^1,$$

$$\tau^b_{e_0e_1} = \tau^b_{e_1e_2} = \cdots = \tau^b_{e_{k-1}e_k} = 0$$

and

$$(e_1, j), (e_2, j), \cdots, (e_k, j)$$

called bow arcs with

$$u^b_{e_1j} = \cdots u^b_{e_kj} = \infty,$$

$$\tau^b_{e_kj} = \tau^1, \tau^b_{e_{k-1}j} = \tau^2, \cdots, \tau^b_{e_1j} = \tau^k.$$

In N^b, the nodes i, j are called the original nodes and $e_1, \cdots, e_k$ are called the artificial nodes. The set of arcs in E^b corresponding to (i, j) is denoted by E^b_{ij}.

Example 4.1. Consider an arc (i, j) with capacity u_{ij} and inflow-dependent transit time $\tau_{ij}(x_{ij})$ as shown in 19(a). Let τ_{ij} be a step function consisting of 3 pieces as shown in Figure 19(b). Corresponding to (i, j), the bow network contains the bow construction as shown in Figure 19(c).

For $e = (i, j) \in E$, let P^e_l be a directed path from $i = e_0$ to e_l consisting of regulating arcs. Any dynamic flow f with time horizon θ in N, can be interpreted as the dynamic flow f^b in the bow network N^b as follows. For $0 \leq \xi < \theta, u^{k-l} < f_{ij}(\xi) \leq u^{k-l+1}$,

$$f^b_{e^b}(\xi) = \begin{cases} f_{ij}(\xi) & \text{if } e^b \in P^e_l \text{ or } e^b = (e_l, j) \\ 0 & \text{otherwise} \end{cases}$$

Example 4.2. In Figure 19, if $f_{ij}(\xi) \in (u^1, u^2]$, then $k - l = 1$, i.e. $l = 2$ (since $k = 3$).

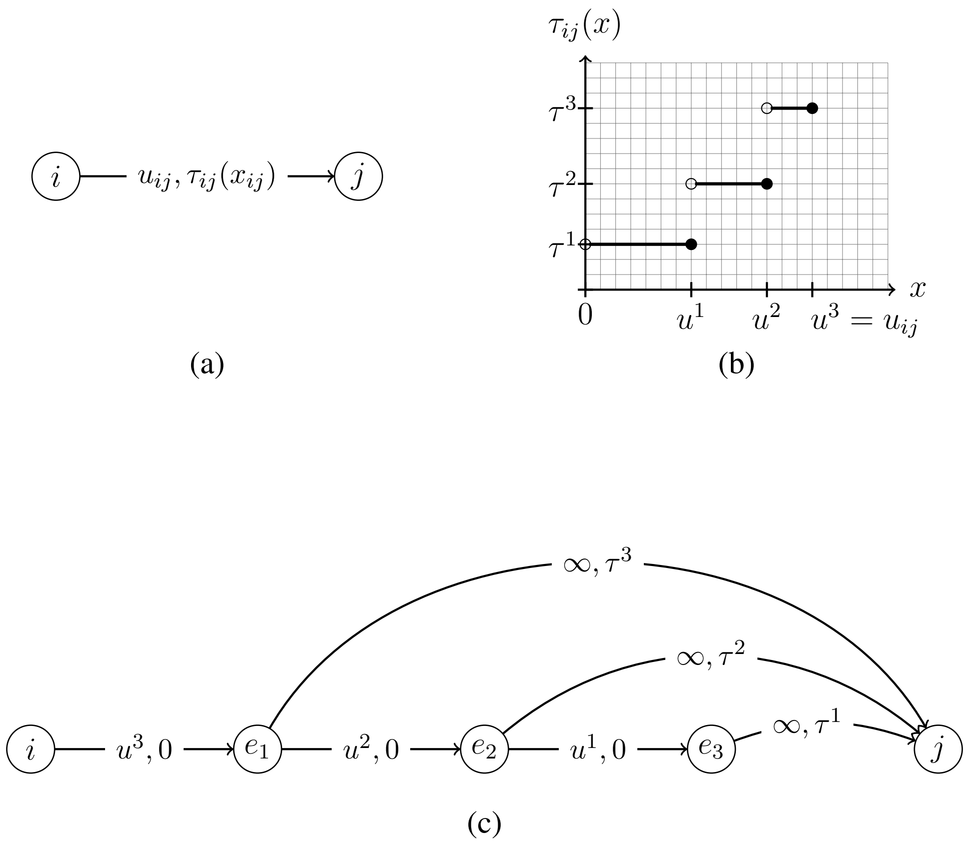


Figure 19: Bow construction of an arc $e = (i, j)$

So,

$$f^b_{ie_1}(\xi) = f^b_{e_1e_2}(\xi) = f^b_{e_2j}(\xi) = f_{ij}(\xi)$$

and

$$f^b_{e_2e_3}(\xi) = f^b_{e_3j}(\xi) = f^b_{e_1j}(\xi) = 0$$

Inflow preserving flow: A dynamic flow in f^b in N^b is called inflow preserving if

i) flow in f^b is stored in the original nodes only,

ii) corresponding to each $(i, j) \in E$, f^b sends flow into at most one bow arc of E^b_{ij}.

In this way, given a flow f with piece-wise constant, non-decreasing and left-continuous inflow-dependent transit time functions on arcs in N which sends Q units of flow from s to t within time θ, there exists a flow f^b with constant transit times in N^b which also sends Q units of flow from s to t. In other words, every dynamic flow in N can be regarded as a dynamic flow in N^b. However, since flow is allowed to split over the bow arcs with different transit times in a bow network, flow units entering (i, j) simultaneously reach the node j at different times in the bow expansion. This is against the assumption of the flow with inflow-dependent transit time. Thus, every flow over time in N^b is not equivalent to a flow over time in N. The conclusion is that dynamic flow in N^b is a relaxation of dynamic flow in N.

4.2.2 Approximate quickest flow with IFDTT

Consider an evacuation network $N = (V, E, u, \tau, s, t)$ with inflow dependent transit time τ. Consider a supply Q at the source s. The dynamic flow (flow over time) corresponding to the minimum time horizon θ that sends Q to the sink t, in such a case, is the quickest flow with inflow-dependent transit time. Köhler, Langkau, and Skutella (2002) have shown that the problem of finding the quickest flow with inflow-dependent transit times is NP-hard (see also Langkau (2003)). Solving the problem in a bow network, they design a 2-approximation algorithm to find an approximate solution. When τ is given as a non-decreasing piece-wise constant left continuous function, the idea of approximation is as follows.

1. Construct a bow graph N^b.
2. Find the static flow x^b corresponding to the temporally repeated quickest flow in N^b. Any of the algorithms discussed in Section 3.2.2 can be used to find such a flow because, the transit time is constant in N^b.
3. For each $e = (i, j) \in E$, find p such that $x^b_{e_pj} > 0$ and $x^b_{e_qj} = 0$ for all $q < p$, and set
$$x_{ij} = x^b_{ie_1}, \tau_{ij}(x_{ij}) = \tau^b(e_{pj})$$
This is done to make the flow inflow-preserving by pushing x^b to the slowest bow arc having the non-zero flow to obtain $\tilde{x}^b$ defined, precisely, as follows:
$$\begin{aligned} \tilde{x}^b_{e_pj} &= \begin{cases} x^b_{ie_1} & \text{if } x^b_{e_pj} > 0 \text{ and } x^b_{e_qj} = 0 \text{ for all } q < p \\ 0 & \text{otherwise} \end{cases}, \\ \tilde{x}^b_{e_{p-1}e_p} &= \begin{cases} x^b_{ie_1} & \text{if } x^b_{e_qj} = 0 \text{ for } q < p \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$
for all $e = (i, j) \in E, 1 \leq p \leq k$ where k is the number of break-points of τ_e.
4. Find the time horizon θ that satisfies $\theta v(x) - \sum_{(i,j)\in E} x_{ij}\tau_{ij}(x_{ij}) = Q$ and obtain f by temporally repeating x.

If a general inflow-dependent (non-decreasing) transit time function is given, it is approximated by a non-decreasing piece-wise left continuous function first and then the above procedure is used.

4.3 Quickest Contraflow with IFDTT

In this section, we discuss the quickest flow with inflow-dependent transit times with a possibility of arc reversals. Given two arcs (i, j) and (j, i) between two nodes i and j, if we reverse (j, i), the capacity of the arc (i, j) is increased by the capacity of the arc (j, i). The flow-dependent transit time functions, generally, depend on the capacity of the arc as well (e.g. BPR function (21), Davidson's function (22)). If the capacity of an arc is increased, more flow can be sent along the arc and the units of flow take less time to travel the same arc. In a contraflow configuration, the auxiliary network is constructed by adding the capacities of the opposite arcs. Therefore the same amount of flow may take less time to reach from one end of the arc to the other end in comparison to the one without contraflow configuration. So, we assume that the transit time τ_{ij} on an arc (i, j) is a function of the inflow rate x_{ij}, the free flow transit time τ^0_{ij}, and the capacity u_{ij}, along with other parameters, e.g. α, β in BPR equation, J in Davidson's equation, which are supposed to be fixed for arcs between i, j. For the contraflow configuration, we assume that the free flow transit time in the two opposite arcs and the arc with which they are replaced with, in the contraflow configuration, are equal. The value of the transit time function on the arc in the auxiliary network is the result of the free flow transit time and the enhanced capacity. Our approach is to find the quickest flow in the form of a temporally repeated static flow x. Let

$$\tau_{ij}(x_{ij}) = g(x_{ij}, \tau^0_{ij}, u_{ij}).$$

Then, for some $x_{ij} = \zeta$, assuming that the free flow transit times on the opposite arcs (i, j) and (j, i) are equal, we have,

$$\tau_{ij}(\zeta) = g(\zeta, \tau^0_{ij}, u_{ij}),$$

$$\tau_{ji}(\zeta) = g(\zeta, \tau^0_{ij}, u_{ji}),$$

and on the auxiliary network

$$\tau'_{ij}(\zeta) = g(\zeta, \tau^0_{ij}, u_{ij} + u_{ji}).$$

We present Algorithm 7 to find the quickest contraflow with inflow dependent transit times in which the transit time is given as a non-negative, non-decreasing, left-continuous, and piece-wise constant function.

Theorem 4.1. *Given a network $N = (V, E, u, \tau, s, t)$ with non-decreasing, left-continuous, piece-wise constant inflow-dependent transit time function. If θ^* is the*

Algorithm 7: Quickest contraflow algorithm with inflow dependent transit times

```
Input  : Evacuation network $N = (V, E, u, \tau, s, t)$, with inflow-dependent
         non-negative, non-decreasing, left continuous, and piece-wise constant
         transit time $\tau$ , and supply $Q$ at $s$
Output: Approximate quickest contraflow
1 Construct the auxiliary network $N' = (V, E', u', \tau', s, t)$ ($\tau'$ is defined as per the
   idea given in the beginning of Section 4.3).
2 Construct the bow network $N'^b$ corresponding to the auxiliary network $N'$(See
   Section 4.2.1).
3 Find the static flow $x^b$ corresponding to the temporally repeated quickest flow in
   $N'^b$.
4 Pushing $x^b$ to the slowest bow arcs in $N'^b$, find the corresponding static flow $x$ in
   $N'$ and the time horizon $\theta^*$ (See Step 3 of the procedure given in Section 4.2.2).
5 Decompose $x$ obtained in Step 4 into chain and cycle flows. Update $x$ by removing
   cycle flows.
6 Reverse $(i, j) \in E$ iff $(j, i) \in E$ and $x_{ji} > u_{ji}$ or $(j, i) \notin E$ and $x_{ji} > 0$.
   Approximate quickest flow in reconfigured $N$ = temporally repeated flow induced
   by $x$ with the time horizon $\theta^*$.
```

quickest time to transship Q units of flow from s to t allowing arc reversals, then a temporally repeated flow with inflow-dependent transit times, allowing arc reversals in N, can be computed in strongly polynomial time, that sends Q units of flow from s to t within a time horizon of at most $2\theta^*$.

Proof. Using Algorithm 7, one can construct such a flow by computing a quickest flow in the bow network N'^b of the auxiliary network N' and then pushing the flow to the slowest arcs. Since the transit time of each arc in N'^b is constant, one can apply any strongly polynomial time algorithm to calculate the quickest flow. The best-known strongly polynomial time algorithm, so far, is the cancel-and-tighten algorithm by Saho and Shigeno (2017) which computes the static flow x^b in N'^b such that

$$\theta^{*b} = \frac{Q + \sum_{(i,j)\in E'^b} \tau_{ij} x^b_{ij}}{v(x^b)}$$

where θ^{*b} is the quickest time to send Q units of flow from s to t in N'^b. If θ^* is the quickest time in the original network, $\theta^* \geq \theta^{*b}$, since N'^b is a relaxation of N'.

Let $\tilde{x}^b$ be the static flow obtained by pushing x^b to the slowest bow arc as described in Step 3 of the procedure given in Section 4.3. So,

$$\theta^{*b} \leq \frac{Q + \sum_{(i,j)\in E'^b} \tau_{ij} \tilde{x}^b_{ij}}{v(\tilde{x}^b)}.$$

But the length of any path $P \in \tilde{\mathcal{P}}$, where $\tilde{\mathcal{P}}$ is the collection of paths in the path decomposition of $\tilde{x}^b$, cannot exceed θ^{*b}. So, from the path decomposition of $\tilde{x}^b$, we can obtain a temporally repeated flow of $\tilde{f}^b$ in N'^b with any time horizon $\theta \geq \theta^{*b}$. Choose θ so that

$$v_\theta(\tilde{f}^b) = \theta v(\tilde{x}^b) - \sum_{(i,j)\in E^b} \tau_{ij}\tilde{x}^b_{ij} = Q$$

Now,

$$\begin{aligned}
v_{2\theta^{*b}}(\tilde{f}^b) &= 2\theta^{*b} v(\tilde{x}^b) - \sum_{(i,j)\in E^b} \tau_{ij}\tilde{x}^b_{ij} \\
&= 2\theta^{*b} v(\tilde{x}^b) - \sum_{P\in\tilde{P}} \tau_P \tilde{x}^b_P \\
&= \theta^{*b} v(\tilde{x}^b) + \sum_{P\in\tilde{P}} (\theta^{*b} - \tau_P)\tilde{x}^b_P \\
&\geq \theta^{*b} v(\tilde{x}^b) \\
&= \theta^{*b} v(x^b) \ \ (\because v(\tilde{x}^b) = v(x^b)) \\
&\geq Q.
\end{aligned}$$

Since v_θ is an increasing function, $\theta \leq 2\theta^{*b}$. As mentioned in the beginning of the proof $\theta^{*b} \leq \theta^*$. Hence, $\theta \leq 2\theta^*$.

The removal of the cycle flows in Step 5 of Algorithm 7 can be performed in $O(mn)$ time and does not change the value of the dynamic flow and Step 6 can be performed in $O(m)$ time. This leads to the assertion of the theorem. □

So far, we have considered step functions as the inflow-dependent travel time functions. However, inflow-dependent travel time functions are, generally, continuous, non-negative, and increasing functions. To solve the problems involving them using bow networks, one has to approximate them by step functions first. We can find the approximate step functions to any desired accuracy by using the following idea given in Langkau (2003).

Approximating a general transit time function by a step function: For $\delta, \eta > 0$ and a non-negative, non-decreasing and left continuous function $\tau_{ij} : [0, u_{ij}] \to \mathbb{R}_{\geq 0}$ with $\tau_{ij}(0) = 0$, a step function τ^{st}_{ij} can be constructed as follows.

1. Choose $\alpha = \lceil \log_{1+\eta}(\tau_{ij}(u_{ij})/\delta) \rceil$.
2. For $k \in \{1, \cdots, \alpha\}$, define $u^k_{ij} = \max\{x : \tau_{ij}(x) \leq (1+\eta)^{k-1}\delta\}$.
3. Define $\tau^{\text{st}}_{ij}(x) = 0$ for $x \in (0, u^1_{ij}]$, and $\tau^{\text{st}}_{ij}(x) = \tau(u^k_{ij})$ for $x \in (u^k_{ij}, u^{k+1}_{ij}], k \in \{1, \cdots, \alpha\}$.

If $\tau_{ij}(0) \neq 0$, one can construct the step function $\bar{\tau}^{\text{st}}_{ij}$ corresponding to a function $\bar{\tau}_{ij} =$

$\tau_{ij}(x) - \tau_{ij}(0), x \in [0, u_{ij}]$, and then $\tau_{ij}^{\text{st}} = \bar{\tau}_{ij}^{\text{st}} + \tau_{ij}(0)$ is the required step function approximation of τ_{ij}.

Clearly, the number of breakpoints of τ_{ij}^{st} is bounded by $\lceil \log_{1+\eta}(\tau_{ij}(u_{ij})/\delta) \rceil + 1$.

Replacing a general inflow-dependent travel time function by the step functions as mentioned above, one can construct a bow network and solve the quickest flow problem in it. Using this idea, Langkau (2003) construct a $(2+\epsilon)$-approximation algorithm to solve the quickest flow problem with inflow-dependent travel times given as general (nonnegative, nondecreasing) functions. Applying the idea in the auxiliary network constructed as described in the beginning of this section leads to:

Theorem 4.2. *There exists a $(2+\epsilon)$-approximation algorithm with strongly polynomial running time for the quickest contraflow problem with inflow-dependent (nonnegative, nondecreasing, left continuous) transit times.*

Moreover, the idea of solving the quickest partial contraflow problem with inflow-dependent transit times is also similar, except that the arcs are reversed proportional to the necessary capacity (See Algorithm 2 steps 4, 5). Hence, we have:

Corollary 4.3. *There exists a strongly polynomial $(2+\epsilon)$-algorithm to solve the quickest partial contraflow problem with inflow-dependent transit times (given as nonnegative, nondecreasing, left continuous functions).*

4.4 Dynamic Flow with LDTT

The total amount of flow on the arc at any time is known as the load of the arc. In this section, we discuss the dynamic flow model with load-dependent transit times (LDTT) . The underlying assumptions of this model are:

i) At each point of time, the entire flow on an arc travels with uniform speed.

ii) The speed depends only on the current load of the arc.

Suppose that the travel time function τ_{ij} depends on the static flow rate x_{ij}, then the load y_{ij} of the arc (i, j) is given by

$$y_{ij} = x_{ij}\tau_{ij}(x_{ij}). \tag{24}$$

Köhler and Skutella (2005) show that if τ_{ij} is monotonically increasing and convex, then x_{ij} is strictly increasing and concave function of the load y_{ij} and express τ_{ij} as a function of load y_{ij} as:

$$\tau_{ij}(x_{ij}) = \hat{\tau}_{ij}(y_{ij}) \tag{25}$$

and present 2-approximation algorithms to solve the quickest flow problem with load dependent transit times realizing its NP-hardness. One of their algorithms solve minimum cost flow problem with convex costs iteratively, and the other solves cost constrained maximum static flow problem, to construct the corresponding dynamic flow. Before presenting the idea of the algorithm, we describe the underlying problems.

4.4.1 Minimum cost (static) flow with convex costs

Given a directed network $N = (V, E)$ the minimum cost flow problem with convex cost is:

$$\min \sum_{(i,j)\in E} c_{ij}(x_{ij})x_{ij} \tag{26}$$

subject to

$$\begin{aligned} \sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} &= b_i \,\forall i \in V & \text{(26a)}\\ 0 \le x_{ij} &\le u_{ij} \,\forall (i,j) \in E & \text{(26b)}\\ x_{ij} &\in \mathbb{Z} \,\forall (i,j) \in E & \text{(26c)} \end{aligned}$$

where b_i denotes the supply associated with $i \in V$, u_{ij}, the (upper) capacity of $(i,j) \in E$, x_{ij} the static flow associated with $(i,j) \in E$, and $c_{ij}(x_{ij})$ is the convex cost depending on the flow x_{ij}. Without loss of generality, the following assumptions are also made:

i) $b_i \in \mathbb{Z} \,\forall i \in V$.

ii) $\sum_{i\in V} b_i = 0$.

iii) $c_{ij}(x_{ij}) = 0$ whenever $x_{ij} = 0$.

If $c_{ij}(x_{ij})$ is piecewise linear convex, one can replace arc (i,j) by parallel arcs corresponding to each piece and cost by the slope of the corresponding piece, to express the minimum cost flow problem with convex costs as an ordinary minimum cost flow problem with linear costs. Allowing each integer point to be a break-point of the function and linearizing the function between these breakpoints, under the integrality assumption, the problem related to any general convex cost function can be dealt with.

Taking an arc corresponding to each piece of the piece-wise linear convex cost function makes the network significantly large. This can be overcome by constructing the residual network N^x corresponding to the static flow x in the following way. If there is an arc (i,j) with $x_{ij} > u_{ij}$, there is an arc in N^x from i to j with cost $c_{ij}(x_{ij} + 1) - c_{ij}(x_{ij})$, and if there is an arc (i,j) with $x_{ij} > 0$, there is an arc in N^x

from j to i with cost $c_{ij}(x_{ij}-1)-c_{ij}(x_{ij})$. The residual capacity of each (i,j) in N^x is set to the maximum flow change for which the unit flow cost remains equal to $c_{ij}(x_{ij}+1)-c_{ij}(x_{ij})$.

In this way, one can apply cycle-canceling algorithm or successive shortest path algorithm both of which run in pseudo-polynomial time.

As in the case of linear costs, one can obtain a polynomial time algorithm by applying capacity scaling approach in the successive shortest path algorithm. A drawback of the successive shortest path algorithm is that it might augment just 1 unit of flow in each flow augmentation so that the number of augmentations is significantly large. The capacity scaling algorithm overcomes this by sending sufficiently large flow in each augmentation to make the number of augmentations is sufficiently small. To apply the algorithm, one needs to construct a Δ-residual network $N^x(\Delta)$ as follows:

i) For each $(i,j)\in E$ with $x_{ij}+\Delta\le u_{ij}$, $N^x(\Delta)$ contains an arc from i to j with a residual capacity Δ and cost $[c_{ij}(x_{ij}+\Delta)-c_{ij}(x_{ij})]/\Delta$.

ii) For each $(i,j)\in E$ with $x_{ij}\ge\Delta$, $N^x(\Delta)$contains an arc from j to i with a residual capacity Δ and cost $[c_{ij}(x_{ij}-\Delta)-c_{ij}(x_{ij})]/\Delta$.

The capacity scaling algorithm for minimum cost flow with convex costs is summarized in the following steps (Ahuja et al., 1993).

1. Initialize $\Delta=2^{\lfloor\log U\rfloor}, x=0, \pi=0$ where $U=\max\{u_{ij}:(i,j)\in E\}$.
2. If any of (i,j) or $(j,i)\in N^x(\Delta)$ violates the reduced cost optimality condition (see Section 3.3.2), increase or decrease the flow x_{ij} by Δ units so that both the arcs satisfy their optimality conditions.
3. $S(\Delta)$ = set of nodes with excesses of at least Δ, $T(\Delta)$ = set of nodes with deficits of at least Δ
4. Unless $S(\Delta)=\varnothing$ or $T(\Delta)=\varnothing$:
    (a) Choose $k\in S(\Delta)$, replace π_i by π_i-d_i for each $i\in V$, where d_i is the length of a shortest k–i path with respect to the reduced costs.
    (b) Identify a shortest k–l (directed) path in $N^x(\Delta)$ with $k\in S(\Delta), l\in T(\Delta)$ and send Δ units of flow from k to l
    (c) Update $x, S(\Delta), T(\Delta), N^x(\Delta)$.
5. Replace Δ by $\Delta/2$.
6. Repeat 2 – 5 unless $\Delta<1$.

4.4.2 Cost constrained maximum static flow

The linear cost constrained maximum static flow with a budget D (Ahuja & Orlin, 1995) is:

$$\max \quad v \tag{27}$$

subject to

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -v & \text{if } i = t \end{cases} \tag{27a}$$

$$0 \leq x_{ij} \leq u_{ij} \; \forall (i,j) \in E \tag{27b}$$

$$\sum_{(i,j)\in E} x_{ij}c_{ij} \leq D \tag{27c}$$

If c_{ij} depends on x_{ij}, then the constraint (27c) looks like

$$\sum_{(i,j)\in E} x_{ij}c_{ij}(x_{ij}) \leq D \tag{27d}$$

If c_{ij} is convex, the corresponding problem is the convex cost constrained maximum static flow problem. Presenting a successive shortest path algorithm (which is a modified version of the successive shortest path algorithm for the minimum cost flow problem with linear costs, Ahuja and Orlin (1995) construct the corresponding capacity scaling algorithm, which can be applied to the convex cost case using the Δ-residual network described in Section 4.4.1. The algorithm runs in a polynomial time.

4.4.3 Quickest flow with LDTT

Given a network $N = (V, E, u, \tau, s, t)$ with load dependent transit times τ and a supply Q at s, the dynamic flow of value Q with minimum time horizon θ is known as a quickest flow with load-dependent transit times.

The first algorithm to solve the problem is based on the fact that a temporally repeated dynamic flow X with load-dependent transit times within time horizon θ can be obtained by solving:

$$\min \quad -v_\theta(X) \tag{28}$$

subject to

$$v_\theta(X) = \theta\left(\sum_{j\in V_t^-} x_{jt} - \sum_{j\in V_t^+} x_{tj}\right) - \sum_{(i,j)\in E} x_{ij}\tau_{ij}(x_{ij}) \tag{28a}$$

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = 0\ \forall i \in V\setminus\{s,t\} \tag{28b}$$

$$0 \le x_{ij} \le u_{ij}\ \forall (i,j)\in E. \tag{28c}$$

As per our assumption each τ_{ij} is increasing and convex. So, problem (28) is a minimum cost flow problem with convex costs and can be solved in polynomial time by using the capacity scaling algorithm discussed in Section 4.4.1. The value of such X is $v_\theta(X)$. However, such a temporally repeated solution may not be a maximum dynamic flow with load-dependent transit times. But for a supply Q at s, if θ^* is the quickest time with load-dependent transit times, then if $\theta = 2\theta^*$, $v_\theta(X) \ge Q$ (Köhler & Skutella, 2005). Embedding this approach into a binary search framework for θ, one gets a $(2+\epsilon)$-approximation for the quickest flow with load-dependent transit times.

The second $(2+\epsilon)$-algorithm solves a cost constrained maximum static flow problem and temporally repeats the flow thus obtained. The procedure is summarized in the following steps.

1. Solve the problem

   $$\max \quad v$$

   subject to

   $$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V\setminus\{s,t\} \\ -v & \text{if } i = t \end{cases}$$
   $$0 \le x_{ij} \le u_{ij}\ \forall (i,j)\in E$$
   $$\sum_{(i,j)\in E} x_{ij}\tau_{ij}(x_{ij}) \le Q$$

   to find the static flow x.

2. Fix the travel time of each $(i,j)\in E$ as $\tau_{ij}(x_{ij})$, using x_{ij} obtained in Step 1.

3. Decompose the flow into directed s–t paths $\mathcal{P}$ and set $\tau_P = \sum_{(i,j)\in P}\tau_{ij}(x_{ij})$.

4. Find minimum time horizon θ such that

$$\sum_{P \in \mathcal{P}: \tau_P \leq \theta} x^P(\theta - \tau_P) = Q.$$

4.5 Quickest Contraflow with LDTT

The problem of finding a quickest flow with load-dependent transit times allowing arc reversals at time zero is known as the quickest contraflow with load-dependent transit times. Using the idea of the auxiliary network construction used to solve the quickest contraflow problem with inflow-dependent transit times, and implementing the algorithms to solve quickest flow problems with load-dependent transit times discussed above in the resulting network, we can find an approximate solution to the quickest contraflow problem with load-dependent transit times. The procedure is given in Algorithm 8.

Algorithm 8: Approximate solution to the quickest contraflow problem with LDTT

```
Input  : $N = (V, E, u, \tau, s, t)$ with a supply $Q$ at $s$ and $\tau$ as a convex function of
         the static flow rate $x$
Output: Approximate quickest contraflow with load-dependent transit times
1 Construct the auxiliary network $N' = (V, E', u', \tau', s, t)$ ($\tau'$ is defined as per the
   idea given in the beginning of Section 4.3).
2 Find the static flow $x$ and time horizon $\theta^*$ corresponding to the approximate
   quickest flow using any of the algorithm described in Section 4.4.3.
3 Decompose $x$ obtained in Step 2 into chain and cycle flows. Update $x$ by removing
   the cycle flows.
4 Reverse $(i, j) \in E$ iff $(j, i) \in E$ and $x_{ji} > u_{ji}$ or $(j, i) \notin E$ and $x_{ji} > 0$.
   Approximate quickest flow in reconfigured $N$ = temporally repeated flow induced
   by $x$ with the time horizon $\theta^*$.
```

Lemma 4.4. *Algorithm 8 runs in polynomial time.*

Proof. The running time of the algorithm is bounded by the running time of Step 2, in which either a minimum cost flow problem with convex costs (Ahuja et al., 1993) or a convex constrained maximum flow problem (Ahuja & Orlin, 1995) is solved. Both the algorithms run in $O((m \log U)S)$ time, where $U = \max\{u_{ij} : (i, j) \in E\}$ and S is the running time a shortest path problem with non-negative costs. Hence Algorithm 8 runs in polynomial time. □

Each of the algorithm presented in Section 4.4.3 is a $(2+\epsilon)$-algorithm. Hence the bound obtained by Algorithm 8 is not worse than that.

Theorem 4.5. *Given transit time on each arc as a convex function of the static flow rate, there exists a $(2+\epsilon)$-algorithm to solve the quickest contraflow problem with load-dependent transit times.*

Proof. It suffices to prove that Algorithm 8 finds a $(2+\epsilon)$-approximate solution to the problem. If θ is the quickest time to send Q units of from s to t in the auxiliary network, then θ^* is obtained by a $(2+\epsilon)$-algorithm in Step 2. So,

$$\theta^* \leq (2+\epsilon)\theta, \; \epsilon > 0$$

Since the removal of cyle flows in Step 3 does not change the value of quickest time, θ^* does not change in this step. Now the quickest flow in the auxiliary network is equivalent to the quickest flow with arc reversals in the original network. Hence the result. $\square$

The modification of the algorithm 8 to the partial contraflow case does not deteriorate the value of the quickest time and the worst-case running time. Hence,

Corollary 4.6. *Given transit time on each arc as a convex function of the static flow rate, there exists a $(2+\epsilon)$-algorithm to solve the quickest partial contraflow problem with load-dependent transit times.*

4.6 Case Illustration

To illustrate some computational results, we consider Kathmandu road network containing major road sections (Figure 20) as an evacuation network N with $n = 44$ and $m = 124$. The transit time (which we consider as the free flow transit time) in each road segment is as provided by Google Maps, and the integer capacity is assumed to be between 1 to 4 flow units per second according to the width of the segment. Related data are given in Appendix A.2

For the purpose of calculating inflow-dependent transit time on each arc (i,j), we consider the BPR function (with $\alpha = 0.15, \beta = 4$) and Davidson's function (with $J = 0.1$) as flow-dependent transit time functions and present analysis corresponding to each of them in parallel.

Given the number of flow units Q to be evacuated, to find the quickest flow allowing (partial) arc reversal, we construct the auxiliary network of the evacuation network. To solve the problem with the transit time depending on the inflow on each arc, we construct the bow graph of the auxiliary network as described in Section 4.2.1.

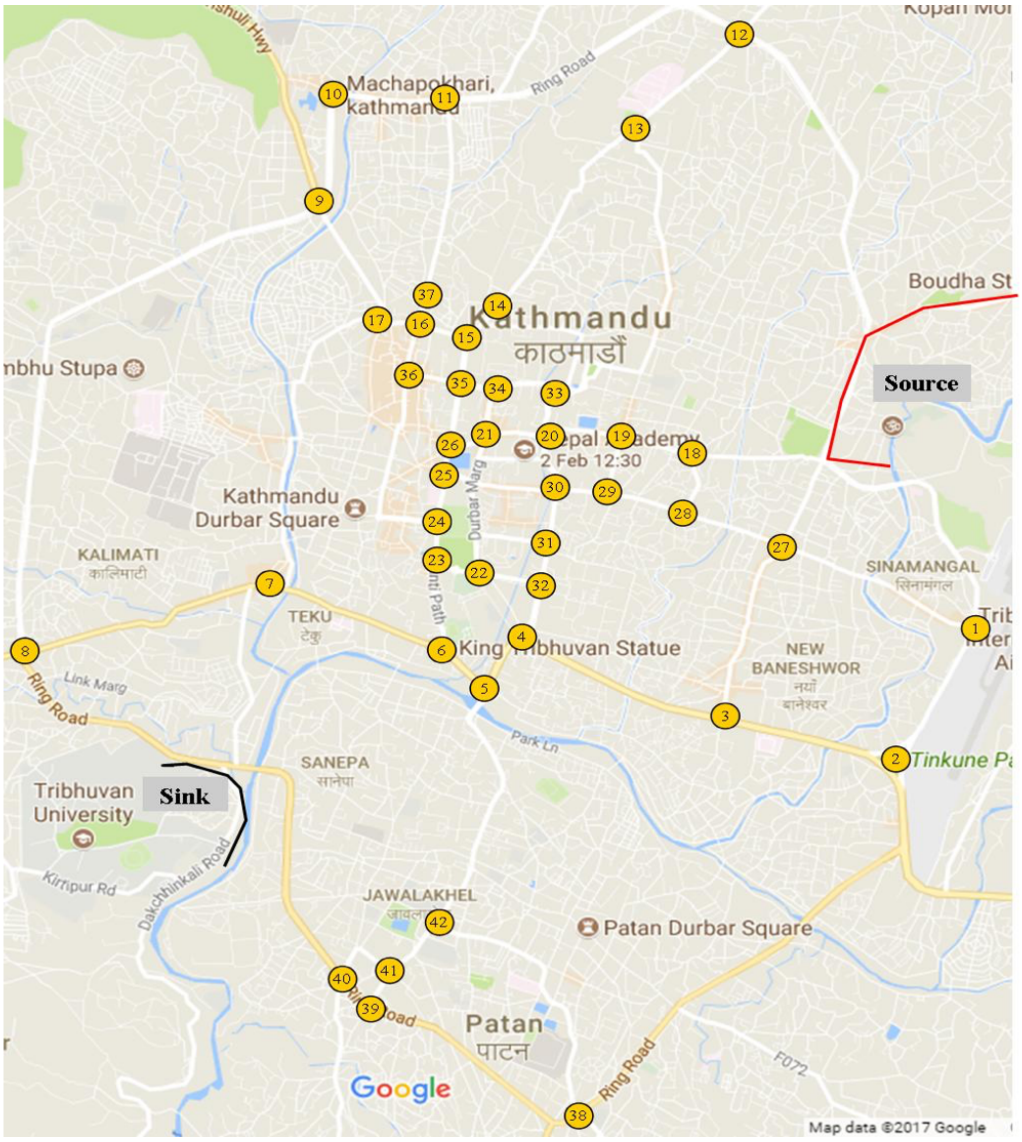
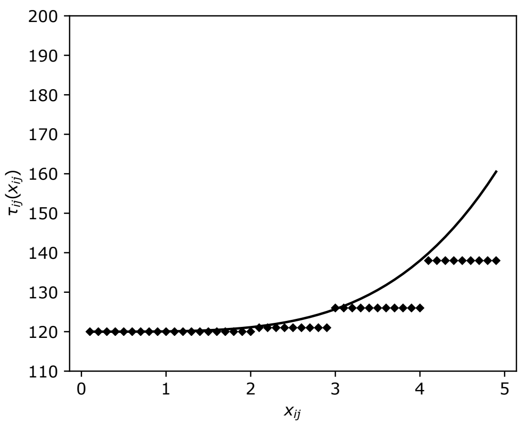
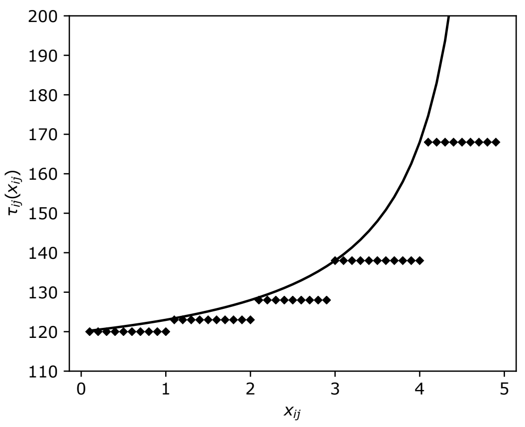


Figure 20: Kathmandu road network.



(a) BPR function



(b) Davidson's function

Figure 21: Flow-dependent transit time functions and corresponding step functions with $\tau_{ij}^0 = 120$ seconds and $u_{ij} = 5$ units per second.

To construct the bow graph, measuring x_{ij}, u_{ij} in flow units per second and t_{ij}^0 in seconds, we consider the transit time function as the step function

$$\tau_{ij}^{\text{st}}(x_{ij}) = \lfloor \tau_{ij}(\lceil x_{ij} \rceil - 1) \rceil, 0 < x_{ij} \leq u_{ij} \tag{29}$$

where $\lceil x_{ij} \rceil$ represents the least integer greater than or equal to x_{ij} and $\lfloor \tau_{ij}(x_{ij}) \rceil$ is the value of $\tau_{ij}(x_{ij})$ rounded to the nearest integer. As an example, the step function representation of a function on an arc with the free flow transit time 120 seconds and capacity 5 units of flow per second is given in Figure 21.

We find the static flow corresponding to the quickest flow in the bow graph, and we push the flow to the slowest arc (see Section 4.2.2) to find the approximate dynamic flow corresponding to the quickest flow. To compare the quickest time θ^* in the bow graph and its approximate value θ^{approx}, after pushing the flow to the slowest blow-arcs, we consider Q between 1 to 10000 with a gap of 500. The results are shown in Figure 23. We find the maximum value of $\frac{\theta^{\text{approx}}}{\theta^*}$ to be 1.045 in case of BPR function and 1.098 in case of Davidson's function.

We compare the quickest times before and after allowing partial arc reversal in Figure 24. For Q as small as 500, the quickest time before allowing (partial) arc reversals using the BPR function is approximately 29.5 minutes; whereas, after allowing arc reversals, it is 27.6 minutes (i.e., approximately 93.5% of the time before allowing arc reversal). With the increase in the value of Q, the gap increases. For Q as large as 100000, the value after allowing arc reversal is 141.7 minutes, 57.6% of the value before allowing arc reversals which is 246.1 minutes. The quickest times for some values of Q before and after allowing arc reversal are listed in Table 3.

Table 3: Comparison of the quickest time before and after allowing arc reversal

	BPR function		Davidson's function	
	Quickest time		Quickest time	
Q_0	Before reversal	After reversal	Before reversal	After reversal
500	29.5	27.6	30.8	28.6
1000	33.6	29.7	35	30.8
10000	58.6	47.4	60.9	49
20000	79.5	58.4	81.8	60.5
50000	142	89.6	144.3	91.7
100000	246.1	141.7	248.4	143.8

Table 4: Number of arcs reversed

	Number of arcs reversed	
Q_0	BPR function	Davidson's function
500	8	5
1000	8	10
10,000	20	21
20,000	29	29
50,000	29	29
100,000	29	29

The number of arcs reversed (partially) for some values of Q are given in Table 4. The observations show that increasing Q beyond a sufficiently large value does not increase the number of arcs reversed beyond some fixed value (e.g., 29 in this case).

The links (arcs) used for the quickest flow corresponding to $Q = 100000$ allowing partial arc reversal (using BPR function and Davidson's function) are depicted in Figure 22, with appropriate direction of the flow. The road segments which need to be reversed fully are (1, Source), (12, Source), (18, Source), (27, Source), (2, 1), (13, 12), (14, 13). (15, 14). (5, 4), (7, 6), (Sink, 8), (17, 16), (16, 15), (Sink, 7), (Sink, 40). The segments which are to be reversed partially are listed in Table 5.

We also compare the quickest times with inflow-dependent transit time on arcs against the quickest times with constant transit time on arcs. For the purpose, we consider three types of constant transit time τ_{ij} for each $(i, j) \in E$:

i) $\tau_{ij} = \tau^{\text{st}}_{ij}(u_{ij})$, the upper bound on the step function represent of $\tau_{ij}(x_{ij})$.

ii) $\tau_{ij} = \bar{\tau}^{\text{st}}_{ij}(x_{ij}) = \frac{\sum_{k=1}^{u_{ij}} \tau^{\text{st}}_{ij}(k)}{u_{ij}}$, the average of the step function values.

iii) $\tau_{ij} = \tau^0_{ij}$ the free flow transit time.

It is observed, in the network considered, that the quickest times corresponding to the constant time on each arc as the average of the corresponding step function are very close to the quickest time with inflow-dependent transit time (Figure 25).

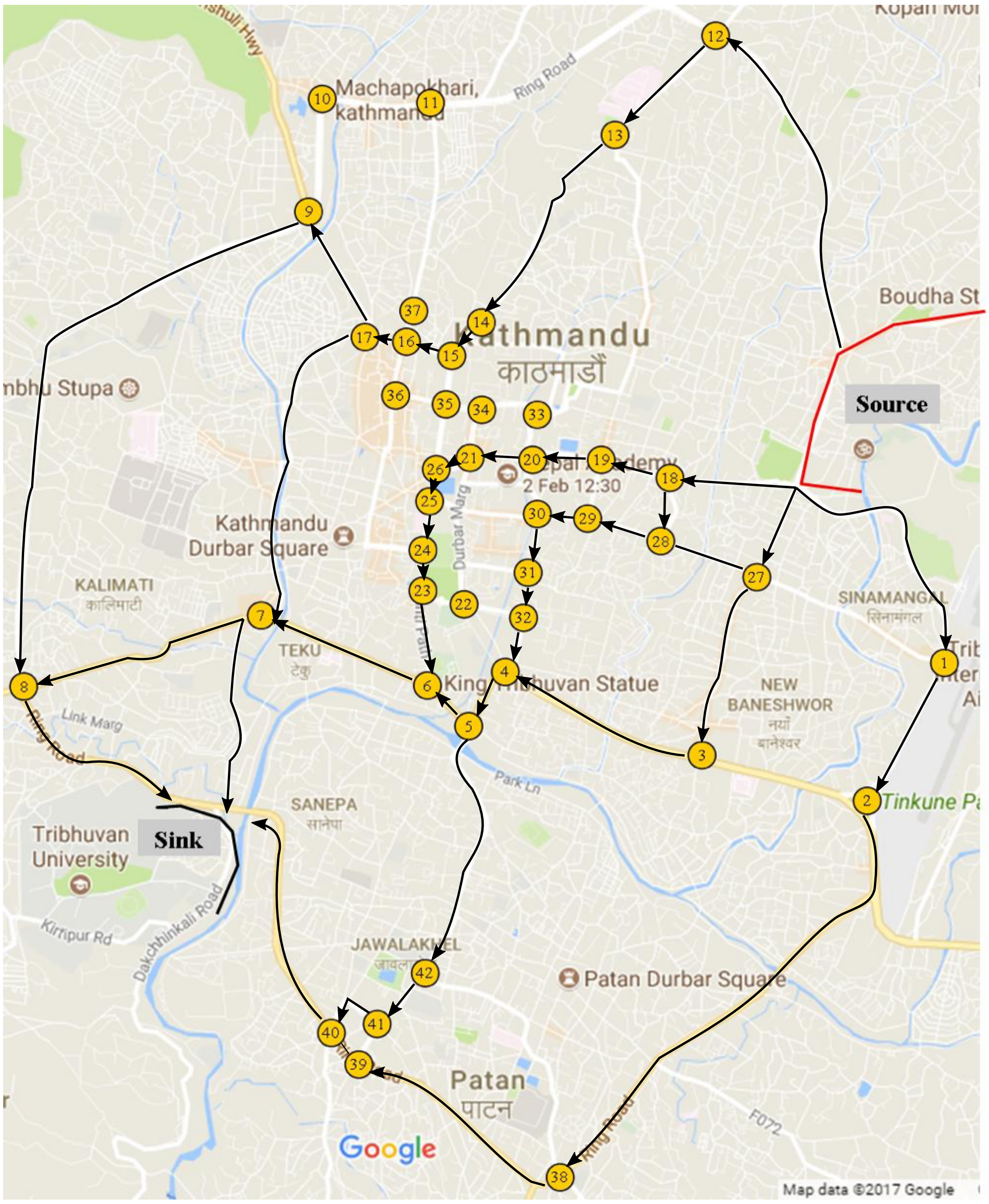
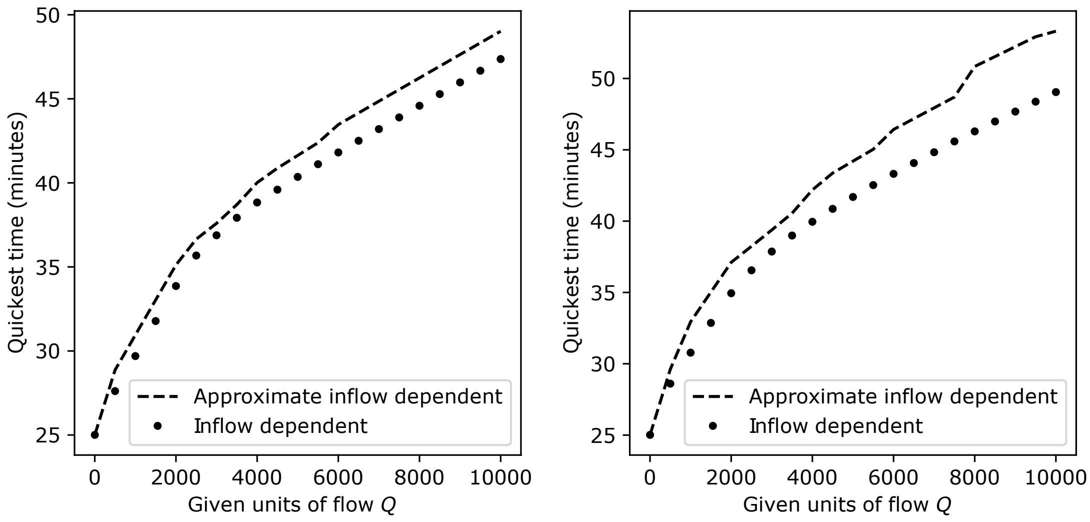


Figure 22: Direction of the approximate quickest flow with arc reversal, $Q = 100000$

Table 5: Partially reversed segments

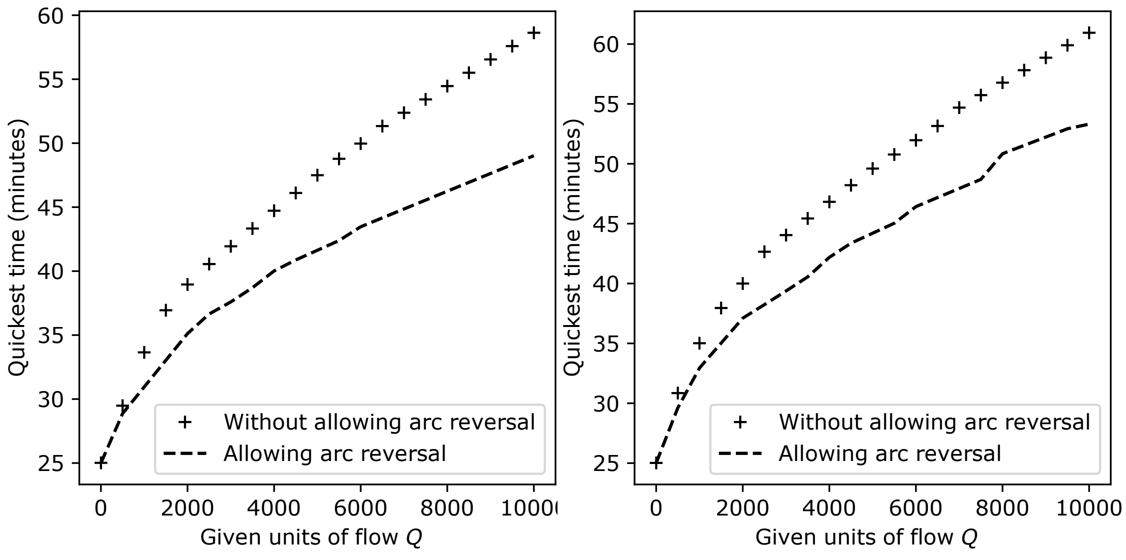
Segment	Reversed capacity	Capacity
(3, 27)	1	2
(38, 2)	1	3
(39, 38)	1	3
(40, 39)	1	3
(6, 5)	1	3
(4, 32)	1	2
(32, 31)	1	2
(8, 7)	1	3
(23, 24)	2	4
(24, 25)	2	4
(26, 21)	2	4
(25, 26)	2	4
(31, 30)	1	2
(19, 18) [a]	1	2
(7, 17) [b]	1	2

[a] for Davidson's function only. [b] for BPR function only.



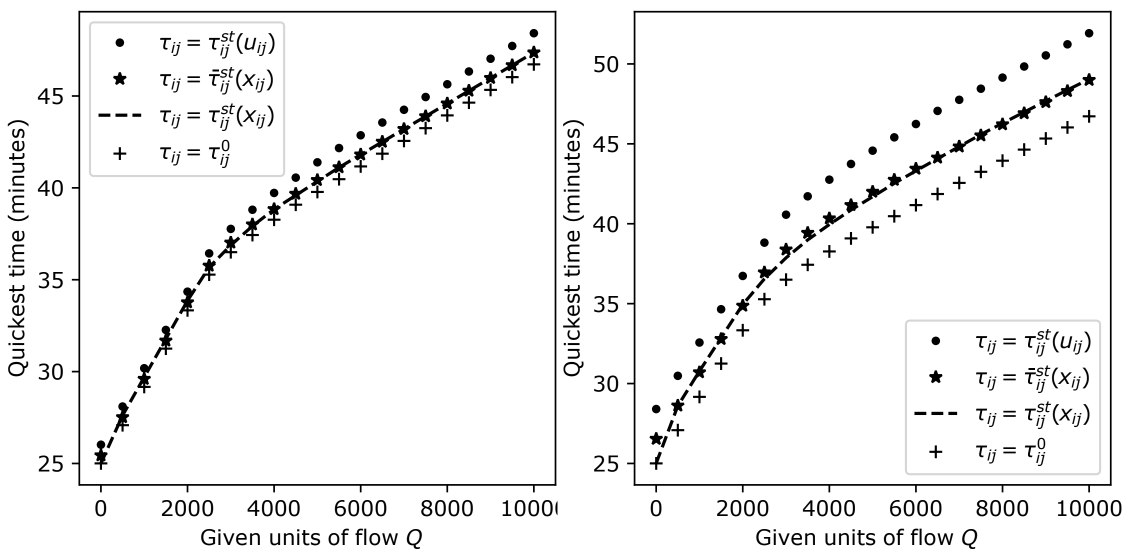
(a) Using BPR function **(b)** Using Davidson's function

Figure 23: Quickest time in bow graph and its approximation



(a) Using BPR function **(b)** Using Davidson's function

Figure 24: Quickest times before and after allowing partial arc reversal



(a) Using BPR function **(b)** Using Davidson's function

Figure 25: Comparison of quickest times (inflow-dependent transit time vs. constant transit time on arcs).

CHAPTER 5

FLOW LOCATION MODELS

5.1 Introduction

If a facility is placed on an arc of a network, it reduces the capacity of the corresponding arc affecting the decisions related to flow. Reduction of the capacity of an arc may result in a reduction in the flow value and an increase in the quickest time for a given amount of flow at a source to reach a sink. Combining location decision problems with network flow problems, Hamacher et al. (2013) model such problems so that there is the least reduction in the maximum flow value (in static, and dynamic cases) because of the placement of the facility. They introduce such problems as network FlowLoc problems, and present efficient algorithms to solve single-facility maximum FlowLoc problems efficiently. Proving that the multi-facility case is NP-hard, they propose polynomial-time heuristics which work quite well in practice.

In this chapter, we extend the ideas to the quickest flow problem to formulate quickest FlowLoc problems (Nath, Pyakurel, & Dhamala, 2018). Mentioning the solution procedures, we realize that the single-facility cases can be solved in strongly polynomial time. We show that the multi-facility case is NP-hard (as in the case of maximum FlowLoc problems) and present efficient heuristics and test their performance taking Kathmandu road network as the evacuation network (Nath, Pyakurel, Dhamala, & Dempe, 2021).

Similarly, We formulate the problem of choosing a sink out of a given set of possible sinks, so as to maximize the flow or minimize the time horizon of the dynamic flow to the sink chosen. We prove that such problems can be solved in strongly polynomial time using a simple search technique.

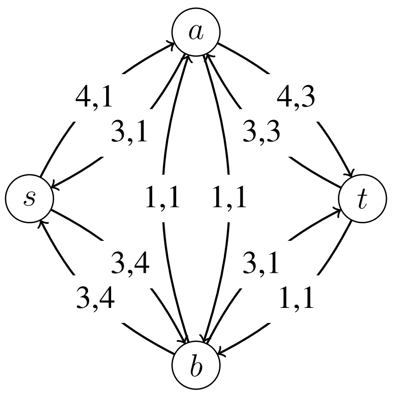


Figure 26: A dummy evacuation network

5.2 Maximum Static/Dynamic FlowLoc

Assigning a facility on any arc of the evacuation network reduces its capacity, and hence there may be a decrease in the maximum static/dynamic flow. Our objective is to find an assignment that minimizes the difference between the maximum static or dynamic flow before and after the assignment.

Definition 5.1 (Maximum static/dynamic FlowLoc)**.** Given a network $N = (V, E, u, \tau, s, t)$, let

(i) $L \subseteq E$ be the set of all feasible locations,

(ii) $\nu : L \to \mathbb{N}$, the number of facilities that can be placed on the possible locations

(iii) F be the set of facilities such that $|F| \leq \sum_{(i,j)\in L} \nu(i, j)$,

(iv) $\sigma : F \to \mathbb{N}$ such that $\sigma(f) \leq \min\{u_{ij} : (i, j) \in L\}, \forall f \in F$, where $\sigma(f)$ is the size of the facility $f \in F$, and

(v) Γ be the collection of allocations $\gamma : F \to L$ of facilities in F to arcs in L such that$|\gamma^{-1}(i, j)| \leq \nu(i, j)\ \forall (i, j) \in L$.

Suppose that v^γ is the value of the maximum static/dynamic s–t flow corresponding to the allocation γ with redefined capacity

$$u^\gamma_{ij} = \begin{cases} u_{ij} - \max\{\sigma(f) : f \in F \text{ and } \gamma(f) = (i, j)\} & \text{if } (i, j) \in \gamma(F) \\ u_{ij} & \text{otherwise} \end{cases} \tag{30}$$

for each $(i, j) \in E$ (see Figure 27). The maximum static/dynamic FlowLoc problem asks for an allocation $\gamma^* \in \Gamma$, such that $v^{\gamma^*} \geq v^\gamma,\ \forall \gamma \in \Gamma$.

Example 5.1. Consider the network depicted in Figure 26. Let there be two facilities f_1, f_2 of sizes 3, and 1 respectively. If they are to be placed in one or both of the arcs (s, a), and (b, s) such that 2 facilities can be placed in (s, a) and only one facility can

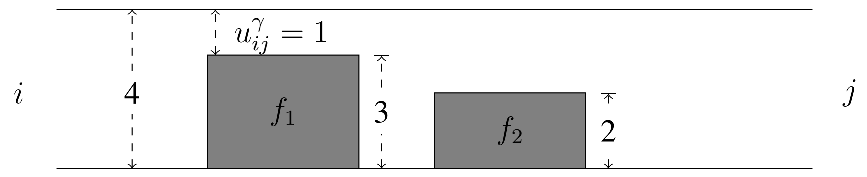


Figure 27: Definition of u_{ij}^{γ}

be placed in (b, s). Here, $L = \{(s, a), (b, s)\}$, $\nu(s, a) = 2$, $\nu(b, s) = 1$, $F = \{f_1, f_2\}$, $\sigma(f_1) = 3$, $\sigma(f_2) = 1$. The collection of all possible allocations is $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$:

$$\gamma_1(f_1) = (s, a), \gamma_1(f_2) = (s, a),$$

$$\gamma_2(f_1) = (s, a), \gamma_2(f_2) = (b, s),$$

$$\gamma_3(f_1) = (b, s), \gamma_3(f_2) = (s, a).$$

For the allocation γ_1, the reduced capacity of (s, a), $u_{sa}^{\gamma_1} = 4 - \max\{3, 1\} = 1$, the capacities of the remaining arcs remain unaltered. The maximum static flow value in this case is $v^{\gamma_1} = 4$. Similarly, $v^{\gamma_2} = 4$ and $v^{\gamma_3} = 6$. Hence $\gamma^* = \gamma_3$. That is, the facility f_1 of size 3 is to be placed in (b, s) and f_2 of size 1 in (s, a).

5.2.1 Single facility maximum static/dynamic FlowLoc problem

The FlowLoc Problem corresponding to $|F| = 1$ is referred to as a single facility FlowLoc problem. Such a problem can be solved by choosing an arc $(i, j) \in L$, reducing its capacity by the size of the facility and solving the corresponding static/dynamic maximum flow problem. The arc which gives the highest value of the maximum flow is the optimal location. We summarize the procedure (Hamacher et al., 2013) in Algorithm 9.

Some modifications can be made to improve the running time of such an algorithm. One idea is to perform the flow calculation at the beginning. Then, either record the remaining capacity $r(i, j)$ for each $(i, j) \in L$ and if there exists $(i^*, j^*) \in L$ such that $\sigma(f) \leq r(i^*, j^*)$, then $\gamma(f) = (i^*, j^*)$, or iterate over elements in L and reroute the flow if possible to calculate the new flow value. It is easy to see that the single-facility maximum FlowLoc problem can be solved in $O(|L|M)$ where M is the complexity of the maximum static or dynamic flow problem. A maximum static flow problem can be solved in strongly polynomial time (see the description before Theorem 3.5). A maximum dynamic flow problem can also be solved in strongly polynomial time by converting it to a minimum-cost circulation problem (see the description before Theorem 3.8). So, the maximum static FlowLoc problem and the maximum dynamic FlowLoc problem can be solved in strongly polynomial time because $|L| \leq n$.

Algorithm 9: Single-facility maximum FlowLoc (Hamacher et al., 2013)

```
Input  : Directed network $N = (V, E, u, \tau, s, t)$ with $F = \{f\}$, $L \subseteq E$
Output: Optimal allocation $\gamma^*$ of $f$
1  curr_max = $-1$
2  for $(i, j) \in L$ do
3      $u_{ij} = u_{ij} - \sigma(f)$
4      max = value of the maximum static/dynamic flow
5      if max > curr_max then
6          curr_max = max
7          $\gamma^*(f) = (i, j)$
8      end
9      $u_{ij} = u_{ij} + \sigma(f)$
10 end
11 return $\gamma^*$
```

Theorem 5.1. *The single-facility maximum (static/dynamic) FlowLoc problem can be solved in strongly polynomial time.*

5.2.2 Multi-facility maximum FlowLoc problems

A FlowLoc problem corresponding to $|F| > 1$ is referred to as a multi-facility FlowLoc problem. The idea of Algorithm 9 can be carried over to this case also. However, such an algorithm iterates over $\binom{|L|}{|F|}$ arcs, even for $\nu(i, j) = 1, \forall (i, j) \in L$. It is a polynomial time algorithm if $|F|$ is fixed, which works good only for small values of $|F|$ or $|L| - |F|$. However, if $|F|$ is the part of the input, the problem is NP-hard. Showing that the decision problem “Does there exist an allocation function $\gamma : F \to L$ such that the maximum static flow value in the network with modified arc capacity u^{γ}_{ij} is greater than or equal to $k \in \mathbb{Z}$?” is NP-complete, Hamacher et al. (2013) establish the following.

Theorem 5.2 (Hamacher et al. (2013))**.** *There is no polynomial time α-approximation algorithm to solve a multi-facility maximum static FlowLoc problem with a finite constant α, unless $P = NP$.*

However, if each of the arcs in L is capable of hosting all the facilities, such a problem can be solved with the complexity of a single-facility FlowLoc problem. Since the facilities in F can be allocated to a single arc in L, we have the following remark.

Remark 5.1. If $|F| \leq \nu(i, j), \forall (i, j) \in L$, and $f^* = \arg\max\{\sigma(f) : f \in F\}$, then $\gamma(f) = \gamma(f^*), \forall f \in F$. Such a problem can be solved by solving the single facility FlowLoc problem with $F = \{f^*\}$.

Table 6: Maximum dynamic FlowLoc decisions (Example 5.2)

θ	v^{γ_1}	v^{γ_2}	γ^*
4	0	1	γ_2
5	4	5	γ_2
6	11	11	γ_1 or γ_2
7	18	17	γ_1

Table 7: Quickest FlowLoc decisions (Example 5.2)

q	θ^{γ_1}	θ^{γ_2}	γ^*
1	4.25	4.00	γ_2
5	5.14	5.00	γ_2
11	6.00	6.00	γ_2 or γ_1
21	7.43	7.67	γ_1

5.3 The Quickest FlowLoc Problem

Definition 5.2 (Quickest FlowLoc)**.** Given a network $N = (V, E, u, \tau, s, t)$, and a supply of Q flow units at the source s, with other considerations as in Definition 5.1, suppose that θ^{γ} is the quickest time to transship the supply the Q units from s to t, corresponding to the allocation γ with redefined capacity u_e^{γ} as in (30). The quickest FlowLoc problem asks for an allocation $\gamma^* \in \Gamma$, such that $\theta^{\gamma^*} \leq \theta^{\gamma}$, $\forall \gamma \in \Gamma$.

Example 5.2. Consider the evacuation network depicted in Figure 26 again. Let $F = \{f\}$, $\sigma(f) = 1$, and $L = \{(a, b), (a, t)\}$. There are two possible allocations, say, γ_1 and γ_2 such that $\gamma_1(f) = (a, b), \gamma_2(f) = (a, t)$. Table 6 shows that the maximum dynamic FlowLoc decisions depend on the time horizon θ. In the *continuous time setting*, when $\theta = 4$, if no facility is placed, a flow of value 1 can reach the sink using only the path s–a–b–t. If the facility is placed on the arc (a, b), this path gets obstructed and no flow can reach the sink, so (a, t) is the optimal location in this case. When $\theta = 5$, if the facility is placed on (a, b), the flow of value 4 can reach the sink via path s–a–t, while the flow of value 5 can reach the sink if we place the facility on (a, t) using the path s–a–b–t with flow value 1 twice and path s–a–t with flow value 3 once. Table 7 shows that the quickest FlowLoc decisions depend on the flow value Q to be transferred from s to t. As expected, the maximum dynamic FlowLoc and the quickest FlowLoc decisions are related with each other.

5.3.1 Single facility quickest FlowLoc problem

As in the case of the single-facility maximum FlowLoc problems, the solution of a single facility quickest FlowLoc problem can be obtained efficiently. We present two algorithms (Algorithm 10 and Algorithm 11) to solve the problem.

Algorithm 10 iterates over all possible arcs $(i, j) \in L$, determines the quickest time if (i, j) hosts the facility and finds the optimal location for the single facility by comparing all those quickest times. It also outputs the quickest flow, and the quickest time after the optimal allocation.

```
Algorithm 10: Single facility quickest FlowLoc I
Input  : Directed network $N = (V, E, u, \tau, s, t)$, a set of feasible locations
         $L \subseteq E, F = \{f\}$, supply $Q$ at the source $s$, size $\sigma(f)$ of the facility $f$
Output: Optimal allocation $\gamma^*$ of $f$, corresponding static flow $x^*$, the
         corresponding quickest time $\theta^*$
1  $\theta^* = \infty$
2  for $(i, j) \in L$ do
3      $u_{ij} = u_{ij} - \sigma(f)$
4      quickest = the quickest time in the modified network
5      if quickest $< \theta^*$ then
6          $\theta^* =$ quickest
7          $\gamma^*(f) = (i, j)$
8          $x^* =$ static flow corresponding to the quickest flow
9      end
10     $u_{ij} = u_{ij} + \sigma(f)$
11 end
12 return $\gamma^*, x^*, \theta^*$
```

Algorithm 10 performs the quickest flow computations $|L|$ times. If we perform a single quickest flow computation before going through Algorithm 10, 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 10. Algorithm 11 addresses this issue.

Theorem 5.3. *The single facility quickest FlowLoc problem can be solved in strongly polynomial time.*

Proof. In the worst case, Algorithm 10 or Algorithm 11 iterates the quickest flow computations $|L|$ times. The cancel-and-tighten algorithm by Saho and Shigeno (2017), 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. As $|L| < m$, the result follows. □

Example 5.3. To illustrate the working of Algorithm 10 and Algorithm 11, we consider the network depicted in Figure 26 with $L = \{(s, a), (s, b), (a, b)\}, \sigma(f) = 1, Q = 11$. In Algorithm 10, we take $\theta^* = \infty$ initially.

```
Algorithm 11: Single facility quickest FlowLoc II
Input  : Directed network $N = (V, E, u, \tau, s, t)$, a set of feasible locations
         $L \subseteq E, F = \{f\}$, supply $Q$ at the source $s$, size $\sigma(f)$ of the facility $f$
Output: Optimal allocation $\gamma^*$ of $f$, corresponding static flow $x^*$, the
         corresponding quickest time $\theta^*$
1 $\theta =$ the corresponding quickest time $(i^*, j^*) = \arg\max\{u_{ij} - x_{ij} : (i,j) \in L\}$
2 if $u_{i^*j^*} - x_{i^*j^*} \geq \sigma(f)$ then
3     $\gamma^*(f) = (i^*, j^*)$
4     $x^* = x, \theta^* = \theta$
5     return $\gamma^*, x^*, \theta^*$
6 else
7     Algorithm 10
8 end
```

First Iteration:
$(i,j) = (s,a)$, $u_{sa} = 4 - 1 = 3$,
quickest = 6.33 $< \infty$, $\theta^* = 6.33$, $\gamma^*(f) = (s,a)$,
$x^*_{sa} = x^*_{ad} = x^*_{sb} = x^*_{bd} = 3$, for the remaining arcs $x^* = 0$ $u_{sa} = 3 + 1 = 4$.

Second Iteration:
$(i,j) = (s,b)$, $u_{sb} = 3 - 1 = 2$, quickest = $6 < 6.33$, $\theta^* = 6$, $\gamma^*(f) = (s,b)$,
$x^*_{sa} = 4, x^*_{ad} = 3, x^*_{sb} = 2, x^*_{bd} = 3, x^*_{ab} = 1$,
$u_{sb} = 2 + 1 = 3$.

Third Iteration:
$(i,j) = (a,b)$, $u_{ab} = 1 - 1 = 0$, quickest = $6 \not< \theta^*$,
$u_{sb} = 2 + 1 = 3$.

Hence the solution is the one obtained in the second iteration. However, in Algorithm 11, we calculate the static flow x associated with the quickest flow at the beginning. The flow x assigned to the arcs in L is

$$x_{sa} = 4, x_{sb} = 2, x_{ab} = 1.$$

We see that $u_{sb} - x_{sb} = 1 \geq \sigma(f)$ implying $\gamma^*(f) = (s,b)$.

5.3.2 Multi-facility quickest FlowLoc problem

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 5.4. *Consider a network $N = (V, E, u, \tau, s, t)$ with a supply $Q > 0$ at s. If $\tau_{ij} = 0 \, \forall (i,j) \in E$, a solution of the quickest flow problem also satisfies the maximum static flow problem, and vice versa.*

Proof. According to Lin and Jaillet (2015), the quickest flow problem can be stated as:

$$\min \quad \frac{Q + \sum_{(i,j)\in A} x_{ij}\tau_{ij}}{v} \tag{31}$$

subject to

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} \quad = \quad \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = t \\ 0 & \text{if } i \notin \{s,t\} \end{cases} \tag{31a}$$

$$0 \quad \leq x_{ij} \quad \leq \quad u_{ij}, \, \forall (i,j) \in E \tag{31b}$$

If $\tau_{ij} = 0 \, \forall (i,j) \in E$, the objective (31) minimizes Q/v and the solution set of the problem becomes equivalent to

$$\max \quad v$$

subject to

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} \quad = \quad \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = t \\ 0 & \text{if } i \notin \{s,t\} \end{cases}$$

$$0 \quad \leq x_{ij} \quad \leq \quad u_{ij}, \, \forall (i,j) \in E$$

which is the maximum static flow problem. □

Theorem 5.5. *There is no polynomial-time α-approximation algorithm to solve the multi-facility quickest FlowLoc problem unless $P = NP$.*

Proof. Suppose that there is a polynomial-time α-approximation algorithm to solve the multi-facility quickest FlowLoc problem for $\alpha < \infty$. According to Lemma 5.4, 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 5.2. □

Because of Theorem 5.5, it is plausible to design a polynomial time heuristic to solve the multi-facility quickest FlowLoc problem. To evaluate the quality of the solution of such a heuristic, first, we present the mixed integer programming formulation of the problem. Using Definition 5.2, and Theorem 3.1, the problem can be formulated as:

$$\min \quad \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v} \tag{32}$$

subject to

$$\begin{aligned}
\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} &= \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = t \\ 0 & \text{if } i \in V \setminus \{s,t\} \end{cases} && \text{(32a)}\\
x_{ij} + \sigma_f y_{ijf} &\le u_{ij},\ \forall (i,j) \in L, f \in F && \text{(32b)}\\
0 \le x_{ij} &\le u_{ij},\ \forall (i,j) \in E && \text{(32c)}\\
\sum_{(i,j)\in L} y_{ijf} &= 1,\ \forall f \in F && \text{(32d)}\\
\sum_{f\in F} y_{ijf} &\le \nu_{ij},\ \forall (i,j) \in L && \text{(32e)}\\
y_{ijf} &\in \{0,1\},\ \forall (i,j) \in L, f \in F. && \text{(32f)}
\end{aligned}$$

The variables and parameters used in the model are described as follows.

Variables

x_{ij} = static flow corresponding to the quickest flow in $(i,j) \in E$

$$y_{ijf} = \begin{cases} 1 & \text{if the facility } f \text{ is placed on } (i,j) \in L \\ 0 & \text{if the facility } f \text{ is not placed on } (i,j) \in L \end{cases}$$

Parameters

Q = given supply at s

σ_f = size of the facility f

u_{ij} = capacity of $(i,j) \in E$

ν_{ij} = number of facilities that can be placed on $(i,j) \in L$

Constraints (32a) and (32c) are conditions for a static flow. Constraints (32b) reduce the capacity of (i,j) by σ_f if the facility f is placed on $(i,j) \in L$. Constraints (32d) state that each facility has to be placed in exactly one arc, and constraints (32e) bound the number of facilities on an arc by the admissible number of facilities on it.

The objective function (32) of the above problem is not linear. To make it linear, we put $1/v = \omega$, and $x_{ij}\omega = \xi_{ij}$. As a result the problem becomes

$$\min \quad Q\omega + \sum_{(i,j)\in E} \tau_{ij}\xi_{ij} \tag{33}$$

subject to

$$\begin{aligned}
\sum_{j\in V_i^+}\xi_{ij}-\sum_{j\in V_i^-}\xi_{ji} &= \begin{cases}1 & \text{if } i=s\\ -1 & \text{if } i=t\\ 0 & \text{if } i\in V\setminus\{s,t\}\end{cases} && \text{(33a)}\\
\xi_{ij}+\sigma(f)\omega y_{ijf} &\le u_{ij}\omega,\ \forall(i,j)\in L, f\in F && \text{(33b)}\\
0\le\xi_{ij} &\le u_{ij}\omega,\ \forall(i,j)\in E && \text{(33c)}\\
\sum_{(i,j)\in L} y_{ijf} &= 1,\ \forall f\in F && \text{(33d)}\\
\sum_{f\in F} y_{ijf} &\le \nu(i,j),\ \forall(i,j)\in L && \text{(33e)}\\
y_{ijf} &\in \{0,1\},\ \forall(i,j)\in L, f\in F && \text{(33f)}
\end{aligned}$$

However, the set of constraints (33b) 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 (1990), replacing (33b) with the following constraints.
$\forall(i,j)\in L, f\in F$,

$$\begin{aligned}
\xi_{ij}+\sigma_f\zeta_{ijf} &\le u_{ij}\omega && \text{(33g)}\\
\zeta_{ijf} &\le My_{ijf} && \text{(33h)}\\
\zeta_{ijf} &\le \omega && \text{(33i)}\\
\zeta_{ijf} &\ge \omega-(1-y_{ijf})M && \text{(33j)}\\
\zeta_{ijf} &\ge 0 && \text{(33k)}
\end{aligned}$$

where M is an upper bound on the value of ω which can be taken 1 if there is at least one path from the source to sink with positive integral capacities and Q is also a positive integer, because v is at least 1 in such cases. So, without loss of generality, putting the objective and the modified constraints together, the problem can be stated as:

$$\min \quad Q\omega+\sum_{(i,j)\in E}\tau_{ij}\xi_{ij} \tag{34}$$

subject to

$$\begin{aligned}
\sum_{j\in V_i^+}\xi_{ij}-\sum_{j\in V_i^-}\xi_{ji} &= \begin{cases}1 & \text{if } i=s\\ -1 & \text{if } i=t\\ 0 & \text{if } i\in V\setminus\{s,t\}\end{cases} && \text{(34a)}\\
\xi_{ij}+\sigma_f\zeta_{ijf} &\le u_{ij}\omega && \text{(34b)}
\end{aligned}$$

$$\omega + y_{ijf} - 1 \leq \zeta_{ijf} \leq y_{ijf} \tag{34c}$$

$$0 \leq \zeta_{ijf} \leq \omega \tag{34d}$$

$$0 \leq \xi_{ij} \leq u_{ij}\omega, \forall (i,j) \in E \tag{34e}$$

$$\sum_{(i,j)\in L} y_{ijf} = 1, \forall f \in F \tag{34f}$$

$$\sum_{f\in F} y_{ijf} \leq \nu(i,j), \forall (i,j) \in L \tag{34g}$$

$$y_{ijf} \in \{0,1\}, \forall (i,j) \in L, f \in F \tag{34h}$$

Now, we present two polynomial time heuristics, in Algorithm 12 and Algorithm 13, to solve the problem. In Algorithm 12, 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) and the residual capacities of the arcs in L are calculated. Then, first $\nu_{i^*j^*}$ facilities are placed on the arc (i^*, j^*) with the largest residual capacity, and (i^*, j^*) 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 (i^*, j^*) is less than the size of the largest facility hosted by it, the quickest flow is recalculated in Line 15.

Example 5.4. To illustrate Algorithm 12, we consider the network given in Figure 26. Let $L = \{(a,s),(a,d),(s,b),(b,t)\}$ with $\nu(a,s) = 1, \nu(a,t) = 2, \nu(s,b) = 1, \nu(b,t) = 3$ and $F = \{f'_1, f'_2, f'_3, f'_4\}$ with $\sigma(f'_1) = 1, \sigma(f'_2) = 3, \sigma(f'_3) = 2, \sigma(f'_4) = 1$.

First of all, we order the facilities in the decreasing order of their size, i.e. $f_1 = f'_2, f_2 = f'_3, f_3 = f'_1, f_4 = f'_4$ so that $\sigma(f_1) = 3, \sigma(f_2) = 2, \sigma(f_3) = 1, \sigma(f_4) = 1$.

The static flow corresponding to the quickest flow is given in the following table. We have denoted $u_{ij} - x_{ij}$ as r_{ij}.

ij	sa	sb	as	at	ab	bs	ba	bt	ta	tb
u_{ij}	4	3	3	4	1	3	1	3	3	1
x_{ij}	4	2	0	3	1	0	0	3	0	0
$r_{ij} : (i,j) \in L$		1	3	1				0		

$\theta = 6, |F| = 4, k = 1 < |F|$.

First iteration:
$(i^*, j^*) = \max\{u_{ij} - x_{ij} : (i,j) \in L\} = (a,s), l = 1, k + l - 1 = 1,$
$\gamma(f_1) = (a,s), L = \{(a,t),(s,b),(b,t)\}, u_{as} = 3 - 3 = 0,$
$u_{i^*j^*} - x_{i^*j^*} + \sigma(f_k) = 0 - 0 + 3 \not< \sigma(f_1),$
$k = 1 + 1 = 2 < |F|$.

Algorithm 12: Multi-facility quickest FlowLoc heuristic I

```
Input  : Directed network $N = (V, E, u, \tau, s, t)$, supply $Q$ at the source $s$, the set
         of possible locations $L$ with number of facilities $\nu : L \to \mathbb{N}$, set of
         facilities $F$ with size $r : F \to \mathbb{N}$
Output: Optimal allocation $\gamma^* : F \to L$, the quickest time $\theta^*$ after the optimal
         allocation
1  Sort the facilities in $F$, according to the size, as $f_1, f_2, \cdots, f_q$ such that
   $\sigma(f_1) \geq \sigma(f_2) \geq \cdots \geq \sigma(f_q)$.
2  $x$ = static flow corresponding to the quickest flow in $N$
3  $\theta$ = the corresponding quickest time
4  $k = 1$
5  while $k \leq q$ do
6      $(i^*, j^*) = \arg\max\{u_{ij} - x_{ij} : (i, j) \in L\}$
7      for $l = 1$ to $l = \nu(i^*, j^*)$ do
8          if $k + l - 1 \leq q$ then
9              $\gamma(f_{k+l-1}) = (i^*, j^*)$
10         end
11     end
12     $L = L \setminus \{(i^*, j^*)\}$
13     $u_{i^*j^*} = u_{i^*j^*} - \sigma(f_k)$
14     if $u_{i^*j^*} - x_{i^*j^*} + \sigma(f_k) < \sigma(f_k)$ then
15         $x$ = static flow corresponding to the quickest flow with modified $u$
16         $\theta$ = the corresponding quickest time
17     end
18     $k = k + \nu(i^*, j^*)$
19 end
20 $\gamma^* = \gamma, x^* = x, \theta^* = \theta$.
21 return $\gamma^*, x^*, \theta^*$
```

Second iteration:
$(i^*, j^*) = \max\{u_{ij} - x_{ij} : (i,j) \in L\} = (a,t)$ (We may take $(i^*, j^*) = (s,b)$ also.)
$l = 1, 2,$
$k + l - 1 = 2 + 1 - 1, 2 + 2 - 1 = 2, 3,$
$\gamma(f_2) = (a,t), \gamma(f_3) = (a,t),$
$L = \{(s,b), (b,t)\},$
$u_{i^*j^*} = 4 - 2 = 2,$
$u_{i^*j^*} - x_{i^*j^*} + \sigma(f_k) = 2 - 3 + 2 = 1 < \sigma(f_k) = 2.$
We recalculate x. The recalculated x is in the following table.

ij	sa	sb	as	at	ab	bs	ba	bt	ta	tb
u_{ij}	4	3	0	2	1	3	1	3	3	1
x_{ij}	3	2	0	2	1	0	0	3	0	0
$r_{ij} : (i,j) \in L$	1						0			

$\theta = 6.4,$
$k = 2 + 2 = 4 = |F|.$

Third iteration:
$(i^*, j^*) = \max\{u_{ij} - x_{ij} : (i,j) \in L\} = (s,b),$
$l = 1,$
$k + l - 1 = 4 + 1 - 1 = 4,$
$\gamma(f_4) = (s,b),$
$L = \{(b,t)\},$
$u_{i^*j^*} = 3 - 1 = 2,$
$u_{i^*j^*} - x_{i^*j^*} + \sigma(f_k) = 2 - 2 + 1 = 1 \not< \sigma(f_i) = \sigma(f_4) = 1,$
$k = 4 + 1 = 5 > |F|.$
Since $k > |F|$, the algorithm terminates and the solution is:
$\gamma^*(f_1') = \gamma(f_3) = (a,t),$
$\gamma^*(f_2') = \gamma(f_1) = (a,s),$
$\gamma^*(f_3') = \gamma(f_2) = (a,t),$
$\gamma^*(f_4') = \gamma(f_4) = (s,b).$

For the optimal assignment, the static flow x^* corresponding to the quickest flow is given in the following table with the quickest time $\theta^* = 6.4$.

ij	sa	sb	as	at	ab	bs	ba	bt	ta	tb
u_{ij}	4	2	0	2	1	3	1	3	3	1
x^*	3	2	0	2	1	0	0	3	0	0

The quickest time θ^* 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 $(i^*, j^*) = (s, b)$ is chosen in the second iteration, the result of the heuristic coincides with that of the MILP solver.

In the above example, if we solve the single-facility quickest FlowLoc problem each time when we require to place a facility in a new arc, the objective function value matches with that of the MILP solution. We present such a procedure to solve a multi-facility quickest FlowLoc problem in Algorithm 13. Although the running time of the algorithm is higher because of more quickest flow calculations, the objective function values are closer to the optimal values in most of the cases in the computational experiment done in Section 5.3.3.

5.3.3 Computational experiment

To test the performance of the proposed heuristics, we take a transportation network of Kathmandu city, consisting of major road segments within the Ring Road (Figure 28(a)). We take Dasharath Stadium and the neighboring area as the source. The stadium is the largest in Nepal and the neighboring area consists of some of the major shopping malls. We consider a scenario of evacuating people in case of an emergency, e.g. the threat of a possible bomb attack, from the source to outside the Ring Road.

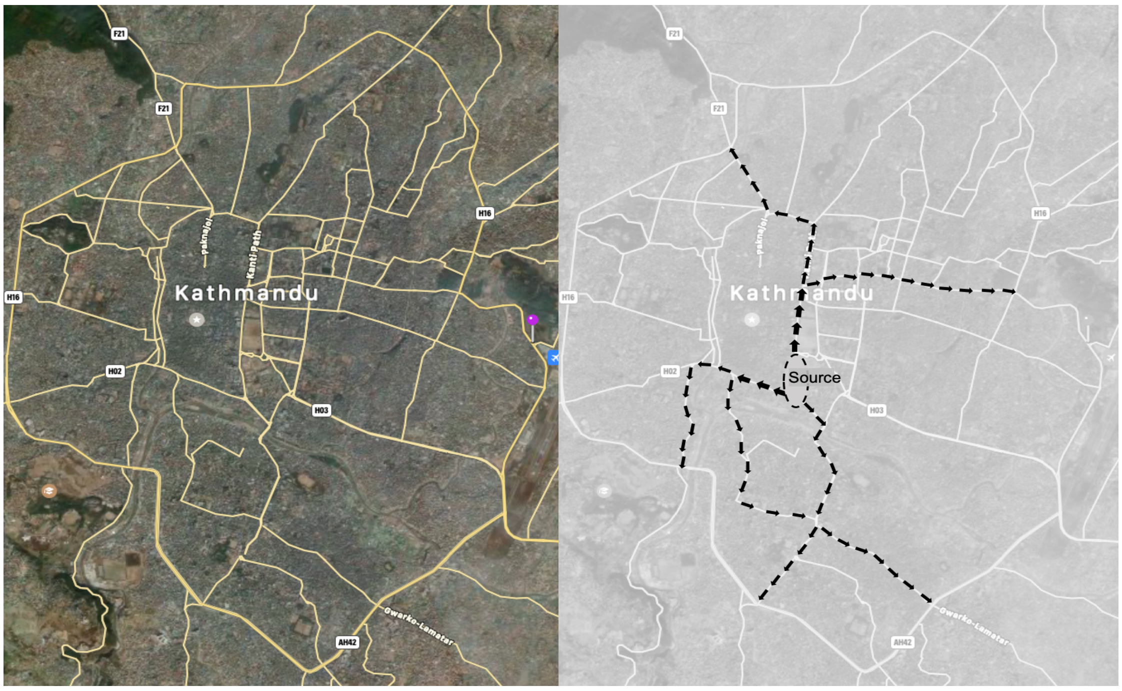
The directed graph representing the network consists of 69 nodes and 221 arcs. The travel time on each arc is considered to be the time of travel on the corresponding road segment at the speed of 40 km/h. The length of each road segment is based on the OpenStreetMap data. Depending on the width of the road segment, the capacity of the corresponding arc is taken from 1 to 4 units of flow (an average sized vehicle) per second.

Given a flow value Q, we calculate the quickest flow and identify the arcs with nonzero flow (see Figure 28(b) for $Q = 20000$) and consider a random set of feasible locations L to contain at least 25% such arcs, because if L consists entirely of zero-flow arcs, the solution is trivial. The set of feasible facilities F is also taken randomly with admissible sizes. For each $(i, j) \in L$, the number of facilities it can host is chosen randomly between 1 to a maximum of 2 facilities per kilometer. We focus, mainly, on the solutions with objective function values differing from the quickest time without allocating facilities.

For $Q = 20000$, outcomes of some typical instances are presented in Table 9. The corresponding quickest time without facility allocation is 2570 seconds. The running time (R. time), and the objective function values (θ^*) are expressed in seconds. The MILP solutions with running time more than 15 minutes are not recorded.

Algorithm 13: Multi facility quickest FlowLoc heuristic II

```
Input  : Directed network $N = (V, E, u, \tau, s, t)$, supply $Q$ at the source $s$, the set
         of possible locations $L$ with number of facilities $\nu : L \to \mathbb{N}$, set of
         facilities $F$ with size $\sigma : F \to \mathbb{N}$
Output: Optimal allocation $\gamma^* : F \to L$, the quickest time $\theta^*$ after the optimal
         allocation
1  Sort the facilities in $F$, according to the size, as $f_1, f_2, \cdots, f_q$ such that
     $\sigma(f_1) \geq \sigma(f_2) \geq \cdots \geq \sigma(f_q)$.
2  $k = 1$
3  while $k \leq q$ do
4      $(i^*, j^*)$ = optimal location obtained by solving the single facility quickest
         FlowLoc problem for $F = \{f_k\}$
5      for $l = 1$ to $l = \nu(i^*, j^*)$ do
6          if $k + l - 1 \leq q$ then
7              $\gamma(f_{k+l-1}) = (i^*, j^*)$
8          end
9      end
10     $L = L \setminus \{(i^*, j^*)\}$
11     $u_{i^*j^*} = u_{i^*j^*} - \sigma(f_k)$
12     $k = k + \nu(i^*, j^*)$
13 end
14 $\gamma^* = \gamma$
15 $x^*$ = static flow corresponding to the quickest flow with modified $u$
16 $\theta^*$ = the corresponding quickest time
17 return $\gamma^*, x^*, \theta^*$
```



(a) Kathmandu network **(b)** Quickest flow arc occupancy ($Q = 20000$)

Figure 28: Quickest FlowLoc case illustration

For $Q = 5000, 20000$ and 50000, with at least 30 instances each (for which the MILP solutions are recorded), the maximum and average percentage deviations from the MILP objective function value are shown in Table 8.

Table 8: Percentage deviation from the MILP objective function values

	Algorithm 12	Algorithm 13
Maximum deviation	21.55%	5.31 %
Average deviation	3.48 %	0.18%

Out of 178 instances (including the above-mentioned instances), only 16 instances of Algorithm 12 have an objective value better than that of Algorithm 13. However, the running time of Algorithm 13 is higher than that of Algorithm 12. Among the tested instances, the maximum running time of Algorithm 12 is 0.59 second with an average of 0.17 second, the corresponding values for Algorithm 13 are 2.31 second and 1.02 second.

The implementation of the algorithms and the mixed integer programming are done using the programming language Python 3.7 on a computer with Mac operating system having 1.8 GHz dual-core Intel Core i5 processor, and 8 GB RAM. The solver used to solve the mixed integer program is CBC (Coin-OR branch and cut).

Table 9: Computational results for some instances with $Q = 20000$

$\|L\|$	$\|P\|$	Algorithm 12		Algorithm 13		MILP	
		R. time	θ^*	R. time	θ^*	R. time	θ^*
5	5	0.14	3209.3	0.53	2717.6	0.87	2713.5
5	7	0.16	2831.9	0.49	2707.6	1.39	2707.1
6	7	0.13	3222.1	0.65	2881.2	1.93	2878.1
8	10	0.15	2580.0	0.58	2575.0	2.92	2575.0
8	8	0.14	3189.3	0.47	3189.3	2.01	3189.3
9	10	0.15	2725.9	0.66	2593.3	3.46	2593.3
10	10	0.17	2587.8	0.73	2585.6	8.01	2585.6
11	10	0.11	2847.5	0.81	2573.9	5.69	2573.9
11	11	0.17	2586.7	0.83	2580.6	15.16	2580.6
11	15	0.20	2721.8	0.72	2710.6	52.52	2707.6
12	11	0.15	2708.2	0.73	2585.0	6.7	2585.0
12	15	0.17	2877.5	0.69	2881.2	114.77	2877.5
13	15	0.12	2588.3	0.51	2576.1	2.82	2576.1
14	15	0.24	2578.3	0.72	2579.4	645.67	2576.1
14	21	0.11	2861.9	0.69	2712.4	134.76	2595.0
15	17	0.16	3005.3	0.98	2843.1	224.36	2843.1
15	20	0.16	3036.7	1.10	2733.5	47.37	2733.5
16	17	0.16	2736.5	1.06	2722.4	221.88	2722.4
17	19	0.19	2851.9	1.29	2717.1	379.54	2716.5
18	19	0.20	2581.7	0.97	2583.3	214.78	2581.1
18	19	0.16	2589.4	1.18	2585.6	822.64	2585.6
19	20	0.24	2595.0	1.93	2607.2	-	-
19	24	0.59	3032.7	1.76	2868.8	-	-
20	25	0.18	2707.6	1.25	2707.6	-	-
21	25	0.15	2577.8	1.21	2577.8	-	-
23	25	0.14	2692.9	1.47	2575.0	-	-
23	26	0.14	2705.3	1.05	2705.3	-	-
24	26	0.20	3026.0	1.87	2863.1	-	-
26	29	0.14	2704.7	1.42	2704.7	-	-
28	40	0.20	2585.0	1.14	2585.0	-	-
28	39	0.17	2586.7	1.20	2586.7	-	-
29	34	0.16	2851.2	1.42	2851.2	-	-
30	40	0.24	2588.3	1.55	2588.3	-	-
31	38	0.17	2851.2	1.75	2851.2	-	-
31	37	0.22	2600.0	1.57	2600.0	-	-
32	40	0.17	2706.5	1.54	2706.5	-	-
37	43	0.22	2701.8	1.77	2701.8	-	-
38	41	0.21	2597.2	1.44	2598.9	-	-
39	50	0.25	2710.0	1.95	2710.0	-	-
40	50	0.09	2573.3	1.76	2571.7	-	-
41	48	0.20	2710.6	2.19	2710.6	-	-

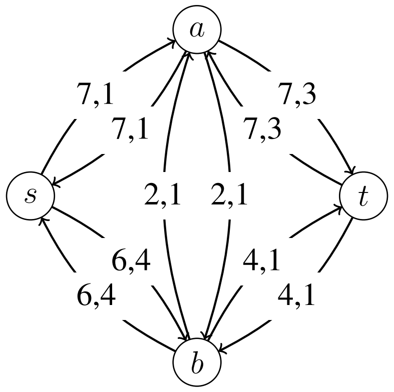


Figure 29: Auxiliary network of the network in Figure 26

5.4 Quickest FlowLoc with Arc Reversals

In this section, we merge the FlowLoc modeling with contraflow. As noted in Chapter 3, if 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), there is a significant reduction in the quickest time for the flow to move from the source to the sink. Arc reversal increases the capacity of the flow in the direction of the arc reversed, and this change in the capacity of arcs, in such cases, has also effects on location decisions.

Analogous to the maximum FlowLoc, we have the following definition.

Definition 5.3 (Maximum static/dynamic ContraFlowLoc)**.** Consider an evacuation network $N = (V, E, u, \tau, s, t)$ with a set of all feasible locations $L \subseteq E$, a set of all facilities F, the size of the facilities $\sigma : F \to \mathbb{N}$ 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 $\gamma : F \to L$, that maximizes static / dynamic flow value, allowing arc reversals.

Corresponding to the quickest FlowLoc problem, we define the quickest ContraFlowLoc problem in the following.

Definition 5.4 (Quickest ContraFlowLoc)**.** Given a network $N = (V, E, u, \tau, s, t)$, a supply Q at s, a set of feasible locations $L \subseteq E$, a set of all facilities F, the size of the facilities $\sigma : F \to \mathbb{N}$, the number of facilities that can be placed on the possible locations and $\nu : L \to \mathbb{N}$, the quickest ContraFlowLoc problem asks for an allocation $\gamma : F \to L$ of the facilities to the arcs, such that the quickest time to transport Q flow units from s to t is minimized, allowing arc reversals.

Example 5.5. Consider the network given in Figure 26. Its auxiliary network is given in Figure 29. The labels on arcs denote capacity and travel time respectively. Let the set of feasible locations $L = \{(a, s), (s, b)\}$ and the size of a single facility f be

$\sigma(f) = 2$. If the facility is placed on (a, s), the value of the maximum static flow before arc reversals is 7 (4 along s–a–t and 3 along s–b–t) and after arc reversals is 11 (5 along s–a–t, 4 along s–b–t, 2 along s–b–a–t). If the facility is placed on (s, b), the corresponding values are 5 (4 along s–a–t, 1 along s–b–t) without arc reversals and 11 (7 along s–a–t, 4 along s–b–t) with arc reversals. Thus the static FlowLoc decision before contraflow configuration is assigning f to (a, s) and after contraflow configuration assigning f to (a, s) or (s, b). The decisions with the quickest time before and after contraflow configuration with $Q = 109$ are listed in Table 10. The optimal location without allowing arc reversals is (a, s), and that allowing arc reversals is (s, b).

Table 10: Quickest time calculations (Example 5.5)

	Quickest time, $Q = 109$	
Facility placed on	Before contraflow	After contraflow
(a, s)	20	15
(s, b)	25.8	14.27
Optimal location	(a, s)	(s, b)

5.4.1 Single-facility quickest ConraFlowLoc

To solve the single-facility maximum static(dynamic) ContraFlowLoc 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 Dhungana and Dhamala (2019)). Here, we present Algorithm 14 to solve the single facility quickest ContraFlowLoc problem, which iteratively chooses an arc from L, reduces its capacity by the size $\sigma(f)$ of the facility f, 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.

We can improve the running time of Algorithm 14 by adapting Algorithm 11 to the contraflow case, in the cases when there is enough capacity in a feasible arc to hold the given facility. After contraflow calculation, if arc $(i, j) \in L$ and its opposite arc $(j, i) \in E$ together have the capacity enough to host the facility, then (i, j) is chosen to locate the facility and we can get rid of the remaining $|L| - 1$ quickest contraflow calculations of Algorithm 14. The procedure is elucidated in Algorithm 15.

In the straight forward procedures mentioned in these algorithms, we basically solve the quickest flow problem iteratively in auxiliary networks and remove cycle flows which requires decomposition of the static flow corresponding to the quickest flow into

Algorithm 14: Single facility quickest ContraFlowLoc I

Input : Directed network $N = (V, E, u, \tau, s, t)$, a set of possible locations L, a supply Q at s, $F = \{f\}$ size $\sigma(f)$ of the facility f

Output: Optimal allocation γ^* of the facility allowing arc reversals, static contraflow x^* corresponding to the quickest contraflow, corresponding quickest time θ^*, set of arcs to be reversed R

```
1  $\theta = \infty$
2  for $(i, j) \in L$ do
3      $u_{ij} = u_{ij} - \sigma(f)$
4      $new\_quickest\_time$ = quickest time in the corresponding auxiliary network
5      if $new\_quickest\_time < \theta$ then
6          $\theta = new\_quickest\_time$
7          $\gamma(f) = (i, j)$
8          $x$ = the corresponding static flow without cycle flows
9      end
10     $u_{ij} = u_{ij} + \sigma(f)$
11 end
12 $\gamma^* = \gamma, x^* = x, \theta^* = \theta$
13 $R = \{(j, i) \in E : x^*_{ij} > u_{ij}$ if $(i, j) \in E$ or $x^*_{ij} > 0$ if $(i, j) \notin E\}$
14 return $\gamma^*, x^*, \theta^*, R$
```

```
Algorithm 15: Single facility quickest ContraFlowLoc II
Input  : Directed network $N = (V, E, u, \tau, s, t)$, a set of possible locations $L$, a
         supply $Q$ at $s$, $F = \{f\}$ size $\sigma(f)$ of the facility $f$
Output: Allocation $\gamma^*$ of the facility allowing arc reversals, static contraflow $x^*$
         corresponding to the quickest contraflow, corresponding quickest time $\theta^*$,
         set of arcs to be reversed $R$
1  $x$ = static flow, without flows in cycles, corresponding to the quickest flow in the
     auxiliary network
2  $\theta$ = the corresponding quickest time
3  $(i^*, j^*) = \arg\max\{u_{ij} + u_{ji} - x_{ij} - x_{ji} : (i, j) \in L\}$
4  if $u_{i^*j^*} + u_{j^*i^*} - x_{i^*j^*} - x_{j^*i^*} \geq \sigma(f)$ then
5      $\gamma(f) = (i^*, j^*)$
6      $u_{i^*j^*} = u_{i^*j^*} - \sigma(f)$
7  else
8      Algorithm 14
9  end
10 $\gamma^* = \gamma, x^* = x, \theta^* = \theta$
     $R = \{(j, i) \in E : x^*_{ij} > u_{ij}$ if $(i, j) \in E$ or $x^*_{ij} > 0$ if $(i, j) \notin E\}$
11 return $\gamma^*, x^*, \theta^*, R$
```

directed paths and cycles. Because the size of the facility does not exceed the capacity of an arc in L (Definition 5.1), and placing the facility on an arc $(i, j) \in L$ reduces the capacity of (i, j) and (j, i) both in the auxiliary network, Algorithm 14 and Algorithm 15 find the location γ with the minimum quickest time. Moreover, the removal of cycle flows in a contraflow computation leads either $x(i, j)$ or $x(j, i)$ to vanish so that the set x^* is feasible in the reconfigured network obtained by reversing the arcs as defined by R. This discussion leads to the following lemma:

Lemma 5.6. *Algorithm 14 or Algorithm 15 solves the single facility quickest ContraFlowLoc problem optimally.*

Theorem 5.7. *The single facility quickest ContraFlowLoc problem can be solved in strongly polynomial time.*

Proof. The complexity of the for loop in Algorithm 14 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 $O(m)$ time, the flow decomposition to remove cycle flows can be done in $O(nm)$ time (Ahuja et al., 1993), the overall complexity of Algorithm 14 is $O(|L|nm^2 \log^2 n)$. Since $|L| \leq m$, the result follows. □

5.4.2 Multi-facility quickest ContraFlowLoc

Replacing quickest flow calculations by maximum static(dynamic) contraflow calculations and adjusting capacities accordingly, Algorithm 12, 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 16.

Further, solving the single-facility quickest ContraFlowLoc problem each time when we require to place a facility in a new arc, we can easily adapt Algorithm 13 to the cotraflow case also.

Example 5.6. To illustrate Algorithm 16, we reconsider the problem illustrated in Example 5.4 with the possibility of arc reversals. The static contra flow corresponding to the quickest contra flow is tabulated in the following table. For simplicity, we write $u_{ij} + u_{ji} - x_{ij} - x_{ji}$ as δ_{ij}.

ij	sa	sb	as	at	ab	bs	ba	bt	ta	tb
u_{ij}	4	3	3	4	1	3	1	3	3	1
x_{ij}	7	2	0	5	2	0	0	4	0	0
$\delta_{ij} : (i,j) \in L$		1	3	1				0		

$\theta = 5.22$, $q = 4$, $k = 1 < q$.

First iteration: $(i^*, j^*) = \max\{\delta_{ij} : (i,j) \in L\} = (s,b)$

$l = 1, k + l - 1 = 1 + 1 - 1 = 1, \gamma(f_1) = (s,b)$,

$L = \{(a,s),(a,t),(b,t)\}$,

$u_{i^*j^*} = 3 - 3 = 0$,

$\delta_{i^*j^*} + \sigma(f_k) = 0 + 3 - 2 - 0 + 3 = 4 \not< \sigma(f_1) = 3$,

$k = 1 + 1 = 2 < q$.

Second iteration: $(i^*, j^*) = \max\{\delta_{ij} : (i,j) \in L\} = (a,t)$,

$l = 1, 2, k + l - 1 = 2 + 1 - 1, 2 + 2 - 1 = 2, 3$,

$\gamma(f_2) = (a,t)$,

$\gamma(f_3) = (a,t)$,

$L = \{(a,s),(b,t)\}$,

Algorithm 16: Multi facility quickest ContraFlowLoc heuristic

```
Input  : Directed network $N = (V, E, u, \tau, s, t)$, a set of possible locations $L$,
         supply $Q$ at the source $s$, set of facilities $F$ with size $\sigma : F \to \mathbb{N}$
Output: Allocation $\gamma^* : F \to L$ allowing arc reversals, the quickest time $\theta^*$ after
         allocation, set of arcs to be reversed $R$
1  Sort the facilities in $F$, according to their size, as $f_1, f_2, \cdots, f_q$ such that
     $\sigma(f_1) \geq \sigma(f_2) \geq \cdots \geq \sigma(f_q)$
2  $x$ = static flow, without cycle flows, corresponding to the quickest flow in the
     auxiliary network
3  $\theta$ = the corresponding quickest time
4  $k = 1$
5  while $k \leq q$ do
6      $(i^*, j^*) = \arg\max\{u_{ij} + u_{ji} - x_{ij} - x_{ji} : (i, j) \in L\}$
7      for $l = 1$ to $l = \nu(i^*, j^*)$ do
8          if $k + l - 1 \leq q$ then
9              $\gamma(f_{k+l-1}) = (i^*, j^*)$
10         end
11     end
12     $L = L \setminus \{(i^*, j^*)\}$
13     $u_{i^*j^*} = u_{i^*j^*} - \sigma(f_i)$
14     if $u_{i^*j^*} + u_{j^*i^*} - x_{i^*j^*} - x_{j^*i^*} + \sigma(f_i) < \sigma(f_i)$ then
15         $x$ = static flow, without cycle flows, corresponding to the quickest flow in
             the auxiliary network with modified $u$
16         $\theta$ = the corresponding quickest time
17     end
18     $k = k + \nu(i^*, j^*)$
19 end
20 $\gamma^* = \gamma, x^* = x, \theta^* = \theta$
21 $R = \{(j, i) \in E : x^*_{ij} > u_{ij}$ if $(i, j) \in E$ or $x^*_{ij} > 0$ if $(i, j) \notin E\}$
22 return $\gamma^*, x^*, \theta^*, R$
```

$u_{i^*j^*} = 4 - 2 = 2,$

$\delta_{i^*j^*} + \sigma(f_k) = 2 + 3 - 5 - 0 + 2 = 2 \not< r(p_2) = 2,$

$k = 2 + 2 = 4 = q$

Third iteration: $(i^*, j^*) = \max\{\delta_{ij} : (i,j) \in L\} = (a, s)$

$l = 1, k + l - 1 = 4 + 1 - 1 = 4,$

$\gamma(f_4) = (a, s),$

$L = \{(b, t)\},$

$u_{i^*j^*} = 3 - 1 = 2,$

$\delta_{i^*j^*} + \sigma(f_k) = 2 + 4 - 7 - 0 + 1 = 0 < \sigma(f_4) = 1$

Recalculation of x:

ij	sa	sb	as	at	ab	bs	ba	bt	ta	tb
u_{ij}	4	0	2	2	1	3	1	3	3	1
x_{ij}	6	2	0	4	2	0	0	4	0	0

$k = 4 + 1 = 5 > q.$

The solution is: $\gamma(f'_1) = \gamma(f_3) = (a, t), \gamma(f'_2) = \gamma(f_1) = (s, b), \gamma(f'_3) = \gamma(f_2) = (a, t), \gamma(f'_4) = \gamma(f_4) = (a, s)$. The static flow x^* corresponding to the quickest flow with arc reversals after this allocation is x as given in the third iteration. The set of arcs reversed before facility allocation is $\{(a, s), (b, a), (t, a), (t, b)\}$ while after allocation it becomes $\{(a, s), (b, a), (t, a), (t, b), (b, s)\}$. 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 Q. Some observations of this problem, after facility-allocation are listed in the following table.

Q	Quickest time	
	Before contraflow	After contraflow
100	24.2	15.33
1000	204.2	115.33
10000	2004.2	1115.33

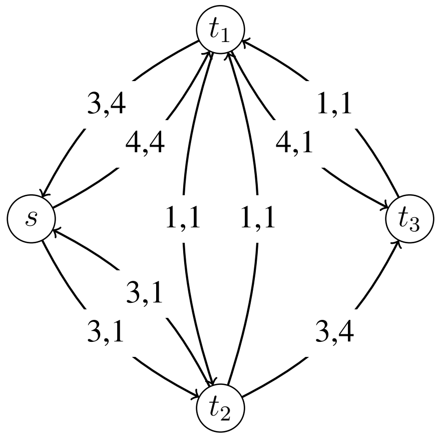


Figure 30: Evacuation network with a source s and possible sinks t_1, t_2, t_3

5.5 Identification of the Optimal Sink

In this section, we extend the idea of FlowLoc developed in the previous sections to identify the sink, from a given set of possible sinks, in a network with a single source. The idea is to choose the sink to optimize the flow (viz. maximize the static/dynamic flow, minimize the time horizon of a given amount of flow). The results of this section are published in Nath and Dhamala (2018).

5.5.1 Optimal sink maximizing the static flow value

Definition 5.5 (MaxStatic sink). Let $N = (V, E, u, \tau, s)$ be a network with a set of possible sinks $T \subseteq V \setminus \{s\}$ and let $v(t)$ denote the value of maximum static s–t flow. We call the node $\arg\max_{t \in T}\{v(t)\}$, the MaxStatic sink.

Example 5.7. We consider an evacuation network in Figure 30. Let the source node be s and the set of feasible sinks $T = \{t_1, t_2, t_3\}$. If node t_1 is taken as the sink, the maximum static flow value $v(t_1)$ is 6 (4 via path s–t_1, 1 each via s– t_2–t_1, and s–t_2–t_3–t_1). Similarly, $v(t_2) = 4, v(t_3) = 7$. By definition, MaxStatic sink is t_3.

Now, we present a mathematical programming formulation to identify the MaxStatic sink. Consider a network $N = (V, E, u, \tau, s)$ with a set of feasible sinks T. For the modeling purpose, we consider that there is only one arc incoming to each $t \in T$. This does not restrict the general case because we can always add a node t' to V, and (t, t') to E such that $u(t, t') = \infty, \tau(t, t') = 0$ and replace $t \in T$ by t' so that t is identified with t'. Practically, infinite capacity of (t, t') can be replaced by $\sum_{i \in V_t^-} u_{it}$. Figure 31 shows such a network transformation in which T becomes $\{t'_1, t'_2, t'_3\}$.

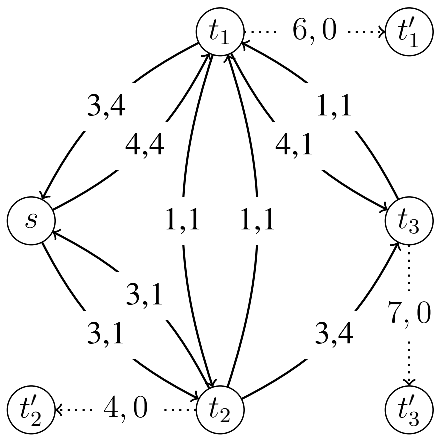


Figure 31: Transformation of the network in Figure 30

Let $E^T = \{(i,t) \in E : t \in T\}$. We present the problem of finding MaxStatic sink as a mixed binary integer formulation:

$$\max \sum_{(i,j)\in E^T} x_{ij} \tag{35}$$

subject to

$$\begin{aligned}
\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} &= 0,\ \forall i \in V \setminus (\{s\} \cup T) && \text{(35a)}\\
x_{ij} &\le u_{ij},\ \forall (i,j) \in E \setminus E^T && \text{(35b)}\\
x_{ij} &\le u_{ij} y_{ij},\ \forall (i,j) \in E^T && \text{(35c)}\\
\sum_{(i,j)\in E^T} y_{ij} &= 1 && \text{(35d)}\\
y_{ij} &\in \{0,1\}\ \forall (i,j) \in E^T && \text{(35e)}\\
x_{ij} &\ge 0\ \forall (i,j) \in E && \text{(35f)}
\end{aligned}$$

The objective (35) maximizes the flow entering to the sinks. Because of (35c)–(35e), the flow will be directed towards only one sink. Constraints (35a) are mass balance constraints at the intermediate nodes. (35b), (35c) are capacity constraints. The constraint (35d) chooses only one sink out of possible sinks.

In the constraints (35c), since u_{ij} may not be in $\{-1, 0, +1\}$, the matrix associated with the constraints is not totally unimodular. Thus the linear programming relaxation of the above integer programming may not give the integral solution. However, if $y_{ij}, (i,j) \in E^T$ is bound to be binary, we have the following observation.

Theorem 5.8. *If $u_{ij} \in \mathbb{Z}\ \forall (i,j) \in E$, the mixed integer programming (35) has all integral solutions.*

Proof. Let $I^{|T|}$ denote the set of the column vectors of the $|T| \times |T|$ identity matrix. Because of our assumption, in-degree of each $t \in T$ is 1, and hence the solution set of the constraints (35d) and (35e) is $I^{|T|}$. If $y = \left[y_{ij} : (i,j) \in E^T\right]$, then the column vector y has exactly $|T|$ components. If a fix $y \in I^{|T|}$ is chosen, the mixed integer programming problem (35) becomes a linear programming problem. Since $|I^{|T|}| = |T|$, MIP (35) can be solved by solving linear program (35)–(35d), (35f) at most $|T|$ times.

Further, let the matrix equation $Mx = 0$ represent the constraints (35a). If (i,j) has both of its ends in $V \setminus (\{s\} \cup T)$, then the column of M corresponding to the variable x_{ij} will have exactly two entries $+1, -1$. Each of the other columns of M corresponding to x_{ij} with an end of (i,j) in $\{s\} \cup T$ will have exactly one entry either 1 or -1. This shows that M is totally unimodular. Since the polyhedron $\{x \in \mathbb{R}^n : Mx = b, 0 \leq x \leq u\}$ with totally unimodular M and integer u is an integral polyhedron, we can get all integer solutions of the mixed integer programming (35) if $u_{ij} \in \mathbb{Z}, \forall (i,j) \in E$. □

The integrality of solutions is particularly important, because most of the network flow algorithms use integrality of flow to develop efficient algorithms.

Although integer programming problems are NP-hard, in general, a straight-forward procedure shows that the problem of finding MaxStatic sink can be solved in strongly polynomial time.

Theorem 5.9. *There exists a strongly polynomial time algorithm to solve the problem of identifying the MaxStatic sink.*

Proof. We present a straight-forward procedure to identify MaxStatic sink in Algorithm 17, which iteratively chooses an element $t \in T$, finds the maximum static s–t flow value, and selects t as the MaxStaic sink if the flow-value is improved. When the iteration ends, the algorithm returns the MaxStatic sink and the corresponding static flow.

Let M be the complexity of a maximum static flow calculation. Since Algorithm 17 terminates after $|T|$ iterations and Line 3 calculates a maximum static flow value in each iteration, the complexity of the algorithm is $O(|T|M)$. The maximum static flow calculation can be done in strongly polynomial time (See the discussion before Theorem 3.5). Hence, Algorithm 17 solves the problem in strongly polynomial time. This proves the assertion. □

Remark 5.2. The practical running time of Algorithm 17 can be improved by finding $v(t)$ in Line 3 only if $\sum_{j \in V_t^-} u_{ij} > curr\_max\_v$ in the network without the transformation mentioned in Figure 31, and exiting the **for** loop and returning t as t^* if $v(t) = \sum_{j \in V_s^+} b_{sj}$.

Algorithm 17: Locating the MaxStatic sink

Input : Network $N = (V, E, u, \tau, s)$, the set of possible sinks T

Output: MaxStatic sink t^*

```
1 $curr\_max\_v = -1$
2 for $t \in T$ do
3     $new\_max\_v = v(t)$
4     if $new\_max\_v > curr\_max\_v$ then
5         $t^* = t$
6         $curr\_max\_v = new\_max\_v$
7     end
8 end
9 return $t^*$
```

5.5.2 Optimal sink maximizing the dynamic flow value

Definition 5.6 (MaxDynamic sink)**.** Let $N = (V, E, u, \tau, s)$ be a network with a set of possible sinks $T \subseteq V \setminus \{s\}$, and a time horizon θ. Let $v_\theta(t), t \in T$ denote the value of the maximum dynamic s–t flow. We call the node $\arg\max_{t \in T}\{v_\theta(t)\}$, the MaxDynamic sink.

Example 5.8. We consider an evacuation network in Figure 30. Let the source node be s and the set of feasible sinks $T = \{t_1, t_2, t_3\}$. If node t_1 is taken as the sink, the maximum static flow value is 6 (4 via path s–t_1, 1 each via s– t_2–t_1, and s–t_2–t_3–t_1). The dynamic flow value with t_1 as the sink and time horizon $\theta = 4$ is 2 (1 via s–t_2–t_1 twice). The values corresponding to other sinks and time horizons are listed in the following table. So, one can easily conclude that the MaxStatic sink is t_3 and the MaxDynamic sink for $\theta = 4$ is t_2, that for $\theta = 9$ is t_1, and for $\theta = 13$ is t_3.

t	$v_4(t)$	$v_9(t)$	$v_{13}(t)$
t_1	2	30	54
t_2	9	28	44
t_3	1	28	56

Since the value of the temporally repeated dynamic s–t flow with a time horizon θ corresponding to a static s–t flow x is $\theta v(x) - \sum_{(i,j)\in E} \tau_{ij} x_{ij}$ where $v(x)$ is the value of x, we can formulate the problem of identification of the MaxDynamic sink for the time horizon θ can be formulated as:

$$\max \ \theta \sum_{(i,j)\in E^T} x_{ij} - \sum_{(i,j)\in E} \tau_{ij} x_{ij} \tag{36}$$

subject to

(35a) – (35f)

Because the constraints set of thie problem is the same as that of the problem 35 The result analogous to that in Theorem 5.8 is valid in this case also. Moreover, such a problem can also be solved in strongly polynomial time.

Theorem 5.10. *The problem of identifying MaxDynamic sink can be solved in strongly polynomial time.*

Proof. Given a time horizon θ, we can easily adapt the procedure in Algorithm 17 to identify the MaxDynamic sink by replacing $v(t)$ by $v_\theta(t)$ in Line 3. For a given sink $t \in T$, let N^t be the network obtained by adding an arc (t, s) to N with $u(t, s) = \infty, \tau(t, s) = -\theta$ (We assume that there is no directed path from t to s in N.) Let x^t be the the min-cost circulation in N^t. Let x be the restriction of x^t in N, then

$$v_\theta(t) = \theta v(x) - \sum_{(i,j)\in E} \tau_{ij} x_{ij}$$

where $v(x)$ is the value of x.

We can calculate $v_\theta(t)$ in a strongly polynomial time because it has been obtained by solving a min-cost circulation problem (e.g. the enhanced capacity scaling algorithm by Orlin (1993) solves it in $O((m \log n)(m + n \log n))$ time). As the mentioned procedure solves such a problem $|T| < n$ times, the MaxDynamic sink can be solved in a strongly polynomial time. □

5.5.3 Optimal sink minimizing the quickest time

Given supply Q at s, recall that the quickest flow problem is to find the dynamic flow which has the value Q within the minimum time horizon. We call such a time horizon as the quickest time. Given a network N, for a fixed Q and a set of possible sinks T, the quickest time depends on $t \in T$. We take a point of view to choose the sink which minimizes the quickest time.

Definition 5.7 (Quickest sink)**.** Let $N = (V, E, u, \tau, s)$ be a network with a set of possible sinks $T \subseteq V \setminus \{s\}$ and let $\theta_Q(t)$ denote the quickest time to transport the flow value of Q from s to $t \in T$. We call the node $\arg\min_{t\in T}\{\theta_Q(t)\}$, the Quickest sink.

We can adapt the mathematical formulation (35), replacing its objective by

$$\frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v}$$

where $v = \sum_{(i,j)\in E^T} x_{ij}$.

The problem of identifying the quickest sink can also be solved in strongly polynomial time.

Theorem 5.11. *There exists a strongly polynomial time algorithm to identify the Quickest sink.*

Proof. A simple procedure mentioned in Algorithm 18 can be used to identify the Quickest sink. The procedure iteratively chooses an element $t \in T$, finds the quickest time to send Q flow units from s to t, and selects t as the Quickest sink if the quickest time decreases. When the iteration ends, the algorithm returns the Quickest sink.

Algorithm 18: Locating the Quickest sink

```
Input  : Network $N = (V, E, u, \tau, s)$, a supply $Q$ at $s$, a set of possible sink
         locations $T$
Output: Quickest sink $t^*$
1 $curr\_quickest\_time = \infty$
2 for $t \in T$ do
3     $new\_quickest\_time = \theta_Q(t)$
4     if $new\_quickest\_time < curr\_quickst\_time$ then
5         $t^* = t$
6         $curr\_quickest\_time = new\_quickest\_time$
7     end
8 end
9 return $t^*$
```

There are $|T|$ iterations in the procedure. In each iteration, we calculate the quickest flow to identify the quickest time. The quickest flow can be calculated by using the algorithm by Saho and Shigeno (2017). Their algorithm uses the idea of cancel-and -tighten algorithm of solving the minimum cost flow problem to compute the quickest flow and runs in $O(nm^2 \log^2 n)$ time. Hence, Quickest sink can be identified in $O(|T|nm^2 \log^2 n)$ time. As $|T| < n$, the result follows. □

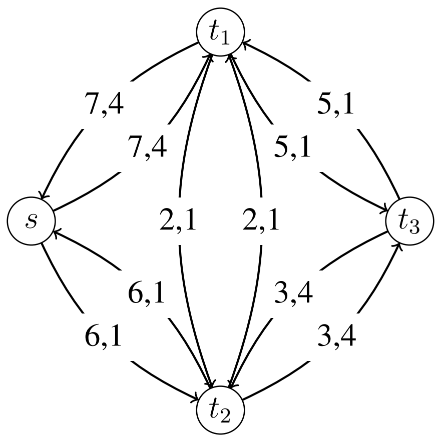


Figure 32: Auxiliary network of the network in Figure 30

5.5.4 Optimal sink with arc reversals

If we allow arc reversals, because of the change in the capacities of the arcs, the decisions related to optimal sink also change. In this section, we develop solution procedures which, not only identify the optimal sink, but also specify which arcs to reverse if the reversal of the arcs is allowed. The idea of the arc reversals developed in the previous chapters is used to identify whether an arc is to be reversed or not.

Example 5.9. Consider the network considered in Example 5.7. To allow arc reversals, we construct the auxiliary network as depicted in Figure 32. Taking s as source, if $t = t_1$, the maximum static flow value is 12 (7 via s–t_1, 3 via s–t_2–t_3–t_1, 2 via s–t_2–t_1). The corresponding values with $t = t_2, t_3$ are 11 and 8, respectively. Thus the MaxStatic sink allowing lane reversals is the node t_1 while it is node t_3 without allowing arc reversals with maximum static flow value 7 (see Example 5.7). Considering t_1 as the sink, the maximum static flow x allowing arc reversals is given in the following table.

ij	st_1	st_2	t_1s	t_1t_3	t_1t_2	t_2s	t_2t_1	t_2t_3	t_3t_1	t_3t_2
x_{ij}	7	5	0	0	0	0	2	3	3	0

Since

$$x_{st_1} > u_{st_1},\ x_{st_2} > u_{st_2},\ x_{t_2t_1} > u_{t_2t_1},\ x_{t_3t_1} > u_{t_3t_1},$$

the arcs to be reversed are: $(t_1, s), (t_2, s), (t_1, t_2)$ and (t_1, t_3).

Given a network $N = (V, E, u, \tau, s)$ and set of possible sinks T, to identify the MaxStatic sink, MaxDynamic sink, Quickest sink allowing arc reversals, we solve the corresponding problem in the auxiliary network N'. We present Algorithm 19 in which the input is $N = (V, E, u, \tau, s)$ and a set of feasible sinks T. For the calculation of the MaxDynamic sink the time horizon θ, and for the calculation of the Quickest sink, the supply Q at the source s are also given. In Line 1, the the auxiliary network

$N' = (V, E', u', \tau', s)$ is constructed. In Line 2, depending on the objective, MaxStatic sink, MaxDynamic sink, or Quickest sink is identified. The procedures described in Section 5.5.1–5.5.3 can be applied for the purpose. The corresponding static flow x is also calculated. In Line 3, x is decomposed into paths and cycles and cycle flows are removed so that the resulting flow is feasible in the reconfigured network after the arc reversals, and the set of arcs to be reversed in Line 4 is well-defined. After the removal of cycle flows, as mentioned previous chapters also, if an arc in E has flow value more than its original capacity, its opposite arc will be reversed. If the solution shows any positive flow in any arc not in E, then the corresponding opposite arc is also reversed. In this way, Line 4 gives the set of arcs to be reversed.

Algorithm 19: Locating sink with contraflow

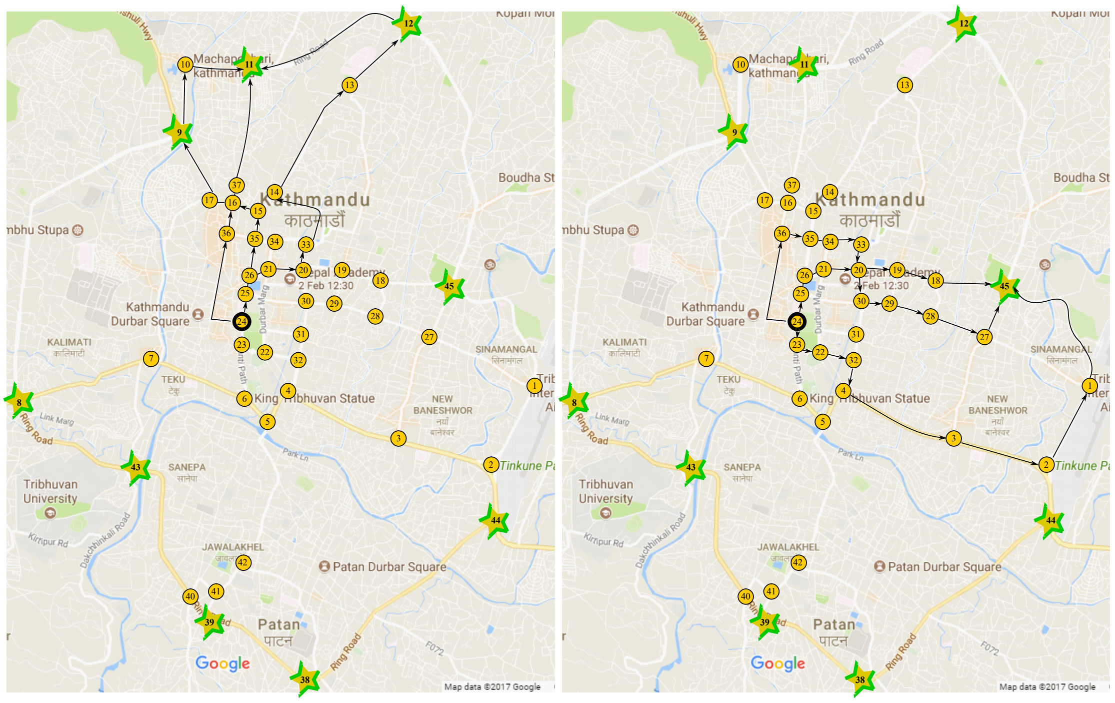
```
Input  : Network $N = (V, E, u, \tau, s)$, the set of possible sink locations $T$ (the time
         horizon $\theta$ in case of the MaxStatic sink, and supply $Q$ at $s$ in case of the
         Quickest sink) with arc reversal capability
Output: Maxstatic/MaxDynamic/Quickest sink $t^*$, and the set $R$ of arcs to be
         reversed
1 Construct the auxiliary network $N'$.
2 Solve the corresponding problem in $N'$ to find the optimal sink $t^*$, and the
   corresponding static flow $x$.
3 Decompose $x$ into paths and cycles and remove the flow in cycles.
4 $R = \{(j, i) \in e : x_{ij} > u_{ij}$ if $(i, j) \in E$ or $x_{ij} > 0$ if $(i, j) \notin E\}$
5 return $t^*, R$.
```

Theorem 5.12. *The problems of identification of MaxStatic sink, MaxDynamic sink, and Quickest sink allowing arc reversals can be solved in strongly polynomial time with the complexity of the corresponding problems without allowing arc reversals.*

Proof. In Algorithm 19, the auxiliary network, in Line 1, can be constructed in $O(m)$ time. Line 2 finds MaxStatic sink, MaxDynammic sink, or Quickest sink in the auxiliary network depending on the problem. So, the complexity of Line 2 is $O(M|T|)$ where M is

(i) the complexity of the maximum static flow calculation, in case of the MaxStatic sink, which can be attained in strongly polynomial time, e.g. the highest label preflow-push algorithm by Goldberg and Tarjan (1988) runs in $O(n^2\sqrt{m})$ time, and the algorithm by Orlin (2013) in $O(mn)$ time if $m < n^{1.06}$,

(ii) the complexity of the minimum cost flow calculation, in case of the MaxDynamic sink. The minimum cost flow problem can be solved in strongly polynomial time,


(a) MaxDynamic sink (Node 11)

(b) MaxDynamic sink with contraflow (Node 45)

Figure 33: MaxDynamic sink with and without contraflow (Kathmandu network)

e.g. the enhanced capacity scaling algorithm of Orlin (1993) runs in $O((m\log n)(m+n\log n))$ time.

(iii) the complexity of quickest flow calculation, which can also be done in strongly polynomial $O(nm^2\log^2 n)$ time (Saho & Shigeno, 2017), in case of the Quickest sink.

The decomposition of a static flow into paths and cycles can be done in $O(nm)$ time (Ahuja et al., 1993). So the time complexity of Line 3 in Algorithm 19 is $O(mn)$. The construction of R in line 4 requires $O(m)$ comparisons. Hence, the overall complexity of the algorithm is dominated by the complexity of Line 2, which is $O(M|T|)$. This proves the assertion. □

5.5.5 Case illustration

As an illustration, we consider Kathmandu road network within Ring Road consisting of the major road segments (Figure 33). We take the node adjoining Tundikhel area (denoted in the figure by 24) as the source and Kalanki (8), Balaju (9), Gongabu (11), Narayan Gopal Chowk (12), Gaushala (45), Koteshwar Jadibuti (44), Satdobato (38), Ekantakuna (39), and Balkhu (43) as possible sinks. To implement auto-based evacuation planning, we take the capacities of arcs between 2 cars per second to 4 cars per second depending on the width of the segment. The direction of the usual traffic

Table 11: Quickest sinks (Kathmandu network)

Q	Without contraflow		With contraflow	
	Quickest sink	Quickest time (sec)	Quickest sink	Quickest time (sec)
1000	Balaju (9)	1160	Balaju (9)	910
10000	Gongabu (11)	3247	Gaushala (45)	2150
20000	Gongabu (11)	4913	Gaushala (45)	2987

flow is taken as the direction of the arc. The time related to an arc is taken using Google Maps.

Considering a time horizon $\theta = 1$ hour, we find the Maxdynamic sink as Gongabu (11) with the dynamic flow value of 12120 cars. However, if we allow arc reversal, the Maxdynamic sink is Gaushala(45) with the corresponding flow value 27360 cars. If we take a time horizon of $\theta = 2$ hours, the sink locations with and without arc reversal are respectively the same as those of $\theta = 1$ hour with flow values 33720 and 70560.

The Quickest sink locations with different values of Q are listed in Table 11.

CHAPTER 6

SAVING A PATH FOR FACILITIES

6.1 Introduction

Using the arc reversal strategy, maximum flow reaching the safe places may be enhanced and the time of evacuation may be reduced. But it may block the paths from a particular node to the source node representing the hazardous area. This obstructs the movement of facilities, if necessary, towards the source. Example 6.1 illustrates this fact.

Example 6.1. Let us consider a network as shown in Figure 34(a) with each arc labeled with its capacity and transit time. Given a time horizon of $\theta = 10$, the maximum dynamic flow from s to t without arc reversals is 29 (1 via the path s–a–b–t repeated 7 times, 2 via s–a–t repeated 6 times, and 2 via s–b–t repeated 5 times). If we allow arc reversals, the value is 57 (1 via the path s–a–b–t repeated 7 times, 5 via s–a–t repeated 6 times, and 4 via s–b–t repeated 5 times). Static flow with arc reversals is shown in Figure 34(b). With the reversal of arcs, there remains no directed path towards the source from any of the remaining vertices.

Movement of facilities, e.g. ambulance, fire-brigade, etc. towards the source may be

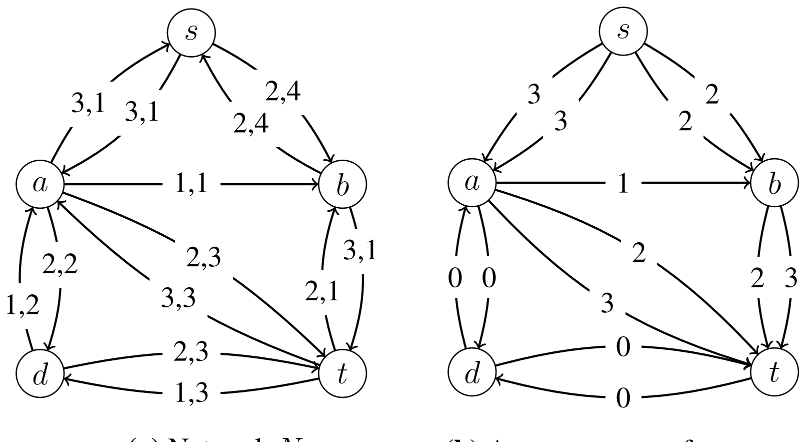


Figure 34: Obstruction of paths towards source because of arc reversals

necessary to save life during the evacuation. The direction of the movement of such facilities may be opposite to the direction of the flow of the evacuees in some of the arcs. Given a depot of facilities, reserving a dedicated path from the depot to the source may have an adverse effect on the flow of the evacuees depending on the choice of the path.

Example 6.2. Let us consider a network in Example 6.1 again. Suppose that s is the source node, t is the sink node and the node d is a depot where the supplies which are to be moved from d to s are located. Consider a time horizon of $\theta = 10$. If the path d–a–s which is of length 3 is to be saved, the value of the maximum dynamic flow with arc reversals is 39, and likewise, if d–t–b–s (length = 8) is saved, the corresponding value is 47. The other values are listed in the following table.

Saved path (P)	$\tau(P)$	$v_{10}(f)$
P_1: d–a–s	3	39
P_2: d–t–a–s	7	39
P_3: d–a–b–s	7	44
P_4: d–t–b–s	8	47
P_5: d–a–t–b–s	10	43
P_6: d–t–a–b–s	11	26

If the objective is to save a path to maximize the dynamic flow regardless of the length of the path chosen, P_4 is optimal. If the length of the path is also to be considered, we may get a different solution. In this chapter, we will look at various possibilities of saving a path.

6.2 Saving a path with a given time bound

When saving a path, there may be situations when the facility from a specified node has to reach the targeted node within a specified time. In those cases, the transit time of the optimal path must not exceed a specified value. In this section, we will consider two problems – one with the objective to maximize the dynamic flow, allowing arc reversals, given a time horizon θ saving a path with transit time bounded by θ_0, and the other with the objective to minimize the time horizon of the given flow value Q in the similar settings. Some of the results in this Section are published in Nath, Pyakurel, Dempe, and Dhamala (2019b).

6.2.1 Saving a path maximizing the dynamic flow

In Example 6.2, if the transit time of the path is to be within 7 units, the optimal saved path to maximize the dynamic flow will be P_3 because the maximum dynamic flow value saving other paths of length less than or equal to 7 units (viz. P_1, P_2) is less than 44.

Let x be a static s–t flow in network $N = (V, E, u, \tau, s, t)$. Given a time horizon θ, the value of the temporally repeated dynamic flow corresponding to x is

$$\theta v(x) - \sum_{(i,j)\in E} \tau_{ij} x_{ij}$$

In the discrete time settings, θ is replaced by $\theta + 1$ (Ford & Fulkerson, 1958; Fleischer & Tardos, 1998; Skutella, 2009).

Hence, the maximum dynamic flow problem, in terms of the static flow to be temporally repeated, can be stated as:

$$\min \quad -\theta v + \sum_{(i,j)\in E} \tau_{ij} x_{ij} \tag{37}$$

subject to

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -v & \text{if } i = t \end{cases} \tag{37a}$$

$$0 \le x_{ij} \le u_{ij}, \forall (i,j) \in E \tag{37b}$$

As described in Chapter 3, the maximum dynamic flow problem, with a possibility of arc reversals in $N = (V, E, u, \tau, s, t)$, is set to find the maximum amount of flow that can be sent from the source s to the sink t if the direction of arcs can be reversed at time 0. The approach finds the static flow x in the auxiliary network $N' = (V, E', u', \tau')$. To ensure the feasibility of such a flow after arc reversals in N, the flows in cycles are removed by decomposing x into directed paths and cycles. An arc in E is reversed if the flow is more than its capacity in the reverse arc of N'. After arc reversals, the temporal repetition of the static flow within the specified time horizon gives the dynamic flow. So, solving the maximum dynamic flow problem (37) in N' may not yield a solution that is feasible in N even after arc reversals. However, if there is no positive flow in the cycles of N', the step of removing flows in cycles can be avoided. In such a case, solving the problem (37) in the auxiliary network will give a feasible optimal solution in N after arc reversals.

If every cycle in the network has the positive time (i.e. the sum of the times on the arcs constituting the cycle is positive), it can be guaranteed that there is no positive cycle flow in a solution of the maximum dynamic flow problem (37).

Theorem 6.1. *If every cycle in N has a positive transit time, an optimal flow in the solution of the maximum dynamic flow problem (37) does not have a positive flow in a cycle.*

Proof. Suppose that $D(x) = -\theta v + \sum_{e \in A} \tau_e x_e$. Let x^* be the optimal solution of the problem (37) with $v = v(x^*) = v^*$ and the flow decomposition of x^* have a positive flow in cycles. Assume $\mathcal{C}$ to be the set of arcs which form a cycle with flow value $\delta > 0$. Define $x^1, x^2 : E \to \mathbb{R}$ by

$$x^1_{ij} = \begin{cases} x^*_{ij} & \text{for } (i,j) \in E \setminus \mathcal{C} \\ x^*_{ij} - \delta & \text{for } (i,j) \in \mathcal{C} \end{cases}$$

and x^2 be a flow defined by

$$x^2_{ij} = \begin{cases} 0 & \text{for } (i,j) \in E \setminus \mathcal{C} \\ \delta & \text{for } (i,j) \in \mathcal{C}. \end{cases}$$

Then x^1 and x^2 are feasible static flows in N such that $x^1_{ij} + x^2_{ij} = x^*_{ij}, \forall (i,j) \in E$. Moreover, since a static flow in a cycle does not contribute to the value of the static flow, $v(x^1) = v^*$. Now,

$$\begin{aligned} D(x^*) &= -\theta v^* + \sum_{(i,j) \in E} \tau_{ij} x^*_{ij} \\ &= -\theta v^* + \sum_{(i,j) \in E} \tau_{ij} x^1_{ij} + \sum_{(i,j) \in E} \tau_{ij} x^2_{ij} \\ &= -\theta v^* + \sum_{(i,j) \in E} \tau_{ij} x^1_{ij} + \delta \sum_{(i,j) \in E} \tau_{ij} \\ &> -\theta v^* + \sum_{(i,j) \in E} \tau_{ij} x^1_{ij} \\ &= D(x^1). \end{aligned}$$

Since x^1 is a also a feasible static flow with the value $v(x^1) = v^*$, this contradicts the optimality of x^*. □

If $\tau_{ij} > 0, \forall (i,j) \in E$, obviously, every cycle in the network N will have positive time. So an immediate corollary is:

Corollary 6.2. *Given $\tau_{ij} > 0, \forall (i,j) \in E$, an optimal flow in the solution of the problem*

(37) does not have a positive flow in a cycle.

If in a network $N = (V, E, u, \tau, s, t)$, $\tau_{ij} > 0 \; \forall (i, j) \in E$, then $\tau'_{ij} > 0$ in the auxiliary network $N' = (V, E', u', \tau', s, t)$. Using Corollary 6.2, the static flow in the solution of the problem (37) in N' will have no positive flow in cycles in its decomposition. As the capacity of each arc $(i, j) \in E'$ in the auxiliary network N' is the sum of the capacities of $(i, j), (j, i) \in N$, we have the following:

Theorem 6.3. *In a network $N = (V, E, u, \tau, s, t)$ with a given time horizon θ, let*

i) for each $(i, j) \in E$, there is $(j, i) \in E$ with $\tau_{ij} = \tau_{ji}$,

ii) for each $(i, j) \in E, \tau_{ij} > 0$.

Then a solution of the linear program

$$\min \quad -\theta v + \sum_{(i,j) \in E} \tau_{ij} x_{ij} \tag{38}$$

subject to

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = t \\ 0 & \text{if } i \in V \setminus \{s, t\} \end{cases} \tag{38a}$$

$$0 \leq x_{ij} \leq u_{ij} + u_{ji}, \forall (i, j) \in E \tag{38b}$$

corresponds to a solution of the maximum dynamic contraflow problem with (i, j) reversed if $x_{ji} > u_{ji}$.

Remark 6.1. In (38), we may replace v by

$$\sum_{j \in V_s^+} x_{sj} - \sum_{j \in V_s^-} x_{js}$$

or by

$$\sum_{j \in V_t^-} x_{jt} - \sum_{j \in V_t^+} x_{tj}.$$

In any case, the constraints (38a) will be replaced by

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = 0 \; \forall i \in V \setminus \{s, t\}.$$

A time-bounded path saving model maximizing dynamic flow: In the light of the above results, we formulate the problem as follows. Consider an evacuation network

$N = (V, E, u, \tau, s, t)$ with the following assumptions:

1. For each $(i, j) \in E$, $\tau_{ij} > 0$.
2. For each $(i, j) \in E$, there exists $(j, i) \in E$ such that $\tau_{ij} = \tau_{ji}$.

If for an arc (i, j), there is no arc (j, i) in the network, we can assume the existence of (j, i) with $u_{ij} = 0$ without loss of generality.

Given a special node d called the depot and a time horizon θ, we consider a situation in which we have to identify a d–s path P of transit time not exceeding θ_0 (i.e. $\tau(P) \leq \theta_0$) such that the dynamic flow is maximized allowing arc reversals except the arcs in P.

The problem is modeled as:

$$\min \quad -\theta \left[\sum_{j \in V_s^+} x_{sj} - \sum_{j \in V_s^-} x_{js} \right] + \sum_{(i,j) \in E} \tau_{ij} x_{ij} \tag{39}$$

subject to:

$$\begin{aligned}
\sum_{j \in V_i^+} y_{ij} - \sum_{j \in V_i^-} y_{ji} &= \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s, d\} \\ 1 & \text{if } i = d \end{cases} && \text{(39a)} \\
y_{ij} &\leq u_{ij}, \forall (i, j) \in E && \text{(39b)} \\
\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} &= 0, \forall i \in V \setminus \{s, t\} && \text{(39c)} \\
0 \leq x_{ij} &\leq (1 - y_{ij}) u_{ij} + (1 - y_{ji}) u_{ji}, \forall (i, j) \in E && \text{(39d)} \\
\sum_{(i,j) \in E} \tau_{ij} y_{ij} &\leq \theta_0 && \text{(39e)} \\
y_{ij} &\in \{0, 1\}, \forall (i, j) \in E && \text{(39f)}
\end{aligned}$$

The objective (39) maximizes the dynamic flow with the time horizon θ. Constraints (39a) specify a d–s path. The binary variable $y_{ij} = 1$ if (i, j) is selected for the path. Constraints (39b) allow only the arcs with positive capacity to be on the path chosen. Constraints (39c) enforce flow conservation at the intermediate nodes. Constraints (39d) limit the flow on (i, j) by (i) its capacity u_{ij} if (j, i) is chosen by the path, (ii) the capacity u_{ji} if (i, j) itself is chosen for the path, and (iii) $u_{ij} + u_{ji}$ if neither (i, j) nor (j, i) is chosen for the path. The constraint (39e) bound the transit time on the path by θ_0.

Remark 6.2. Some important notes on the above model are:

i) The path constraints (39a) may not always give a simple d–s path (as shown in

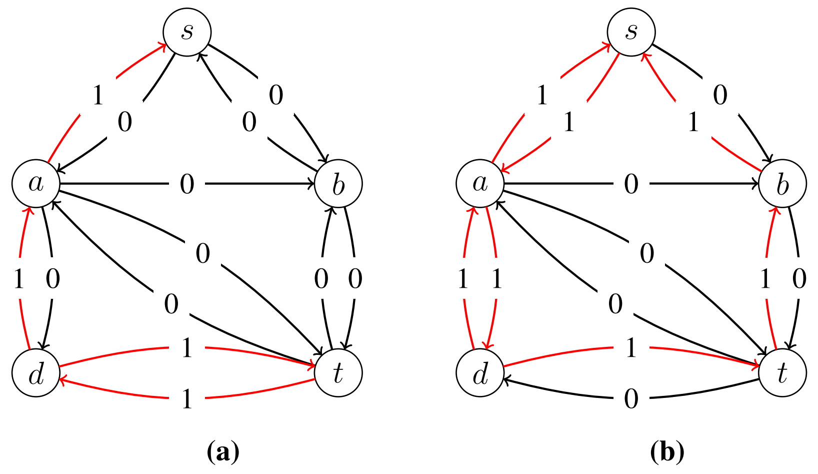


Figure 35: Path constraints (39a) may not yield a simple d–s path

Figure 35. We will accept such a path also in the solution because it always contains a simple path to be used for practical purpose. Moreover, removal of cycles in such a path does not deteriorate the objective function value.

ii) For feasibility, there must be at least one d–s path with transit time θ_0.

iii) The model is valid even if s in the constraints (39a) is replaced by any other node in $d' \in V$ to save a d–d' path.

iv) Each arc of the path saved by the above model will have capacity at least one. However, if one needs a path with a capacity at least $b \in \mathbb{Z}_{>0}$, one can define

$$U_{ij} = \begin{cases} 1, & b \le u_{ij} \\ 0, & b > u_{ij} \end{cases}$$

and replace the constraint (39b) by

$$y_{ij} \le U_{ij}, \forall (i,j) \in E \tag{40}$$

This will force $y_{ij} = 0$ when $b > u_{ij}$. Consequently, the arcs with capacity less than b will not be chosen for the saved path.

Theorem 6.4. *Given $\tau_{ij}, u_{ij}, \theta \in \mathbb{Z}_{\ge 0}$, there exists a solution of the problem (39), in which $x_{ij} \in \mathbb{Z}_{\ge 0}\ \forall (i,j) \in E$.*

Proof. Choosing y to satisfy (39a), (39b), and (39f), the problem is converted to a maximum dynamic flow problem in the auxiliary network of the network obtained by removing a path from d to s. Since, the maximum dynamic flow problem satisfying the hypothesis has an integral solution, the result follows. □

6.2.2 Saving a path minimizing the time horizon

Example 6.3. In Example 6.1, the minimum time of sending a flow of 57 units from s to t allowing arc reversals is 10. If we save the path d–a–s, the minimum time increases to 12.57. The corresponding times saving the available paths are given in the following table.

Saved path (P)	$\tau(P)$	Quickest time
P_1: d–a–s	3	12.57
P_2: d–t–a–s	7	12.57
P_3: d–a–b–s	7	11.63
P_4: d–t–b–s	8	11.25
P_5: d–a–t–b–s	10	11.75
P_6: d–t–a–b–s	11	16.20

The minimum time can be computed by removing the path considered and solving a quickest flow problem allowing arc reversals in the resulting network. For the purpose of solving the quickest flow problem with arc reversals, we need to solve the quickest flow problem in the auxiliary network and decompose the static flow thus obtained in the chains and cycles and remove the cycle-flows. Analogous to Theorem 6.1, we have,

Theorem 6.5. *If every cycle in N has a positive transit time, an optimal flow in the solution of the quickest flow problem (18) does not have a positive flow in a cycle.*

Proof. Let x be a static flow s–t flow N. Suppose that

$$\theta(x) = \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v(x)}$$

Let x^* be the optimal solution of (18), on page 22, with $v = v(x^*) = v^*$ and the flow decomposition of x^* have a positive flow in a cycle. Assume $\mathcal{C}$ to be the set of arcs which form a cycle with value $\delta > 0$. Define x^1, x^2 as in the proof of Theorem 6.1. Then

$$\begin{aligned}
\theta(x^*) &= \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x^*_{ij}}{v(x^*)} \\
&= \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x^1_{ij} + \delta \sum_{(i,j)\in E} \tau_{ij}}{v(x^1)} \\
&> \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x^1_{ij}}{v(x^1)}
\end{aligned}$$

contradicting the optimality of x^*. $\square$

An immediate corrollary is:

Corollary 6.6. *Given $\tau_{ij} > 0, \forall (i,j) \in E$, an optimal flow in the solution of the problem (18) does not have a positive flow in a cycle.*

Because of Theorem 3.1 and the Algorithm 1, analogus to Theorem 6.3, we have:

Theorem 6.7. *In a network $N = (V, E, u, \tau, s, t)$ with a supply Q at s, let*

i) for each $(i,j) \in E$, there is $(j,i) \in E$ with $\tau_{ij} = \tau_{ji}$,

ii) for each $(i,j) \in E, \tau_{ij} > 0$.

Then a solution of the linear program

$$\min \quad \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v} \tag{41}$$

subject to

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = \begin{cases} v & \textit{if } i = s \\ -v & \textit{if } i = t \\ 0 & \textit{if } i \in V \setminus \{s,t\} \end{cases} \tag{41a}$$

$$0 \leq x_{ij} \leq u_{ij} + u_{ji}, \forall (i,j) \in E \tag{41b}$$

corresponds to a solution of the quickest contraflow problem in which (i,j) reversed if $x_{ji} > u_{ji}$.

A time-bounded path saving model minimizing the time horizon: With the assumptions made in the path saving model (39), except that in stead of the time horizon θ, a supply of Q flow units is given at the source, and the objective is to minimize the time-horizon of the dynamic flow with value Q. We model the problem as:

$$\min \quad \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v} \tag{42}$$

subject to:

$$\sum_{j \in V_i^+} y_{ij} - \sum_{j \in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,d\} \\ 1 & \text{if } i = d \end{cases} \tag{42a}$$

$$\begin{aligned} y_{ij} &\leq u_{ij}, \forall(i,j) \in E && \text{(42b)} \\ \sum_{j\in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} &= \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -v & \text{if } i = t \end{cases} && \text{(42c)} \\ 0 \leq x_{ij} &\leq (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \forall(i,j) \in E && \text{(42d)} \\ \sum_{(i,j)\in E} \tau_{ij} y_{ij} &\leq \theta_0 && \text{(42e)} \\ y_{ij} &\in \{0,1\},, \forall(i,j) \in E && \text{(42f)} \end{aligned}$$

With an assumption that there is at least one directed s–t path of a finite transit time, we can linearize the objective by putting

$$\frac{1}{v} = \omega, \frac{x_{ij}}{v} = \xi_{ij}$$

so that (42) becomes $Q\omega + \sum_{(i,j)\in E} \tau_{ij}\xi_{ij}$, and the constraints (42c), (42d) become

$$\sum_{j\in V_i^+} \xi_{ij} - \sum_{j \in V_i^-} \xi_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -1 & \text{if } i = t \end{cases} \tag{42g}$$

$$0 \leq \xi_{ij} \leq (1-y_{ij})\omega u_{ij} + (1-y_{ji})\omega u_{ji}, \forall(i,j) \in E. \tag{42h}$$

Further, using the idea given in Torres (1990), we put $(1-y_{ij})\omega = \zeta_{ij}$ in (42h) so that we get the following linear constraints, $\forall(i,j) \in E$.

$$\begin{aligned} \zeta_{ij} &\geq \omega - Uy_{ij} && \text{(42i)} \\ \zeta_{ij} &\geq L(1-y_{ij}) && \text{(42j)} \\ \zeta_{ij} &\leq \omega - Ly_{ij} && \text{(42k)} \\ \zeta_{ij} &\leq U(1-y_{ij}) && \text{(42l)} \end{aligned}$$

where U and L are, respectively, an upper bound and a lower bound of ω. Assuming Q to be a positive integer, and there is at least one s–t path with a finite transit time, the value of v is at least 1, and hence $0 < \omega \leq 1$. With this assumptions, we replace L by 0, U by 1, and the problem can be stated as:

$$\min \quad Q\omega + \sum_{(i,j)\in E} \tau_{ij}\xi_{ij} \tag{43}$$

subject to:

$$\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,d\} \\ 1 & \text{if } i = d \end{cases} \tag{43a}$$

$$y_{ij} \leq u_{ij}, \forall (i,j) \in E \tag{43b}$$

$$\sum_{j\in V_i^+} \xi_{ij} - \sum_{j\in V_i^-} \xi_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -1 & \text{if } i = t \end{cases} \tag{43c}$$

$$0 \leq \xi_{ij} \leq \zeta_{ij} u_{ij} + \zeta_{ji} u_{ji}, \forall (i,j) \in E \tag{43d}$$

$$0 \leq \zeta_{ij} \leq \omega, \forall (i,j) \in E \tag{43e}$$

$$\omega - y_{ij} \leq \zeta_{ij} \leq 1 - y_{ij}, \forall (i,j) \in E \tag{43f}$$

$$\sum_{(i,j)\in E} \tau_{ij} y_{ij} \leq \theta_0 \tag{43g}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{43h}$$

According to Lin and Jaillet (2015), the quickest flow problem has an integer optimal solution if the inputs are integral. Hence, analogous to Theorem 6.4, we have

Theorem 6.8. *Given $\tau_{ij}, u_{ij}, Q \in \mathbb{Z}_{\geq 0}$, there exists a solution of the problem (42), in which $x_{ij} \in \mathbb{Z}_{\geq 0}\ \forall (i,j) \in E$.*

The problems (39), and (43) are mixed integer linear programs. Various methods, e.g. branch-and-bound, branch-and-cut, branch-and-price algorithms are available for solving such problems. Various commercial and non-commercial software solvers are available to implement the algorithms. One such implementation is presented in the following section.

6.2.3 Case Illustration

We consider a road network of Kathmandu inside Ring Road considering only the major road-segments (shown in Figure 36). The network consists of 44 nodes, and 132 arcs. We take Pashupati Nath region as a source, where a large gathering of people takes place in various religious occasions, and Tribhuvan University region as a sink, where there is a sufficient open space. For a car-based evacuation planning, we assume the capacity of each segment from 2 cars per second to 4 cars per second according to the width of the segment. The travel time to traverse the segment is taken as provided by Google Maps data. We consider node 24 as the depot node d which adjoins Tundikhel region from where medical facilities, support from army, etc. can be sent towards the

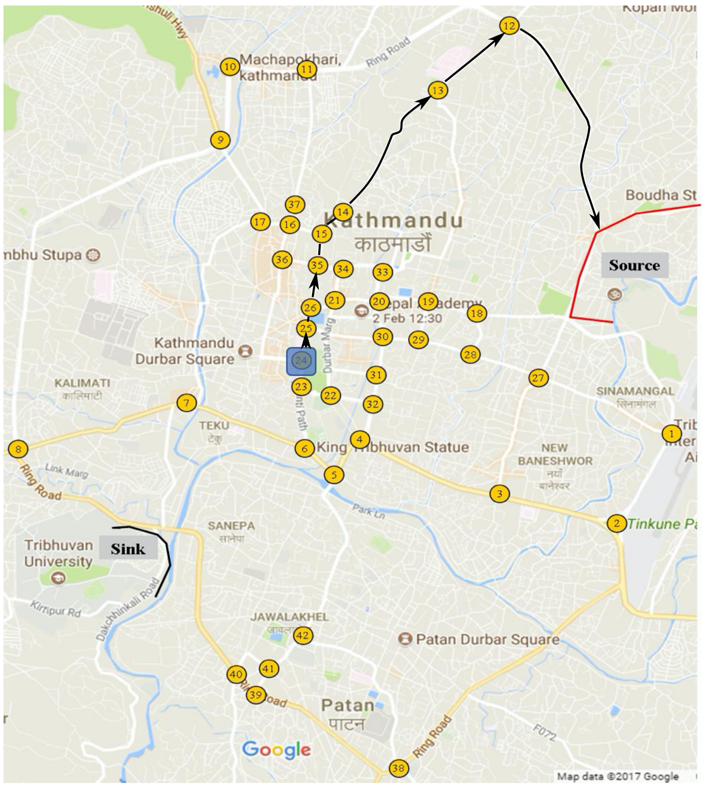


Figure 36: Time-bounded saved path, case illustration

source. Considering a time horizon $\theta = 1$ hour and $\theta_0 = 30$ minutes, we find that 21000 cars can be evacuated without saving the path 24–25–26–21–20–33–13–12–source. If we consider $\theta = 2$ hours and $\theta_0 = 1$ hour, the corresponding results are are 71400 and 24–25–26–35–15–16–37–11–12–source respectively.

With $Q = 100000$ and $\theta_0 = 30$ minutes, the quickest time is found to be 154 minutes with the saved path 24–25–26–35–15–14–13–12–source. With $Q = 50000, \theta_0 = 30$ minutes, the corresponding results are 94.52 minutes and 24–25–26–21–34–33–14–13–12–source.

Using CPLEX 12.6.3 solver, the solutions has been obtained within 0.1 seconds in a computer with Intel Core i5, 2.30 GH processor, 4GB RAM, and 64-bit Windows operating system.

6.3 A Bicriteria Optimization Approach

6.3.1 Multicriteria optimization

Let $\psi_i : \mathcal{X} \to \mathbb{R}, i = 1, \cdots, p$, then a problem of the form

$$\begin{aligned} \min \quad & (\psi_1(x), \cdots, \psi_p(x)) \\ \text{subject to} \quad & x \in \mathcal{X} \end{aligned} \tag{44}$$

where $\mathcal{X}$ is called the feasible set. The minimization (min) has not be considered in the ordinary sense because elements of $\mathbb{R}^p$ are not ordered in the sense the elements in $\mathbb{R}$ are ordered. We define some special orders in $\mathbb{R}^p$ as follows.

Componentwise order in $\mathbb{R}^p$: For $a = (a_1, \cdots, a_p), b = (b_1, \cdots, b_p) \in \mathbb{R}^p$, the following orders in $\mathbb{R}^p$ are defined.

i) $a \leqq b \Leftrightarrow a_1 \leq b_1, \cdots, a_p \leq b_p$ (weak componentwise order)

ii) $a \leq b \Leftrightarrow a_1 \leq b_1, \cdots, a_p \leq b_p, a \neq b$ (componentwise order)

iii) $a < b \Leftrightarrow a_1 < b_1, \cdots, a_p < b_p$ (strict componentwise order)

Pareto optimality, efficiency, weak efficiency: Let $\psi = (\psi_1, \cdots, \psi_p)$. For $a, b \in \mathcal{X}$, if $\psi(a) \leq \psi(b)$, we say that a dominates b, and $\psi(a)$ dominates $\psi(b)$. A feasible solution $a^* \in \mathcal{X}$ is called Pareto optimal or efficient if there is no $a \in \mathcal{X}$ that dominates a^*. In other words, a^* is Pareto optimal if there is no $a \in \mathcal{X}$ satisfying $\psi(a) \leq \psi(a^*)$. A feasible solution $a^w \in \mathcal{X}$ is called weakly Pareto optimal (weakly efficient) if there is no $a \in \mathcal{X}$ satisfying $\psi(a) < \psi(a^w)$.

It is obvious that an efficient solution is also a weakly efficient solution, but the converse is not true. The following result is important to find the solutions of the bi-criteria model developed in this work.

Theorem 6.9 (Ehrgott (2005)). *An optimal solution of the problem*

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & \psi_j(x) \\ \text{subject to} \quad & \psi_k(x) \leq \epsilon_k, k = 1, \cdots, p \quad k \neq j \end{aligned} \tag{45}$$

where $\epsilon \in \mathbb{R}^p$, is a weakly efficient solution of the problem (44).

6.3.2 Bicriteria path-saving model maximizing dynamic contraflow

Example 6.4. In Example 6.1, if we consider only the maximum flow value, the path P_4 is the optimal path with a maximum flow of value 47. But sometimes, the length of the path also matters, e.g. to move the facility the fastest. The possible candidate for this option is the path P_1. Considering the objectives of minimizing the path length and maximizing flow value, the non-dominated paths are P_1, P_3, and P_4.

Motivated by above example, we consider a bicriteria problem that minimizes the length of the saved path and maximizes the dynamic flow allowing arc reversals in the remaining network (Nath, Dempe, & Dhamala, 2021).

Consider a network $N = (V, E, u, \tau, s, t)$ with a depot node d, in which $(j, i) \in A$

whenever $(i, j) \in A$, and $\tau_{ij} = \tau_{ji}$. We formulate the problem of identification of the shortest path maximizing the dynamic flow with arc reversals as a bicriteria optimization problem on the basis of the famous shortest path problem formulation and Theorem 6.3.

Assuming that

$$\psi_1 = \sum_{(i,j)\in E} \tau_{ij} y_{ij}$$

and

$$\psi_2 = -\theta \left[\sum_{j\in V_s^+} x_{sj} - \sum_{j\in V_s^-} x_{js} \right] + \sum_{(i,j)\in E} \tau_{ij} x_{ij},$$

the problem is formulated as:

$$\min \ (\psi_1, \psi_2) \tag{46}$$

subject to:

$$\begin{aligned}
\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} &= \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s, d\} \\ 1 & \text{if } i = d \end{cases} && \text{(46a)}\\
y_{ij} &\leq u_{ij}, \forall (i,j) \in E && \text{(46b)}\\
\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} &= 0, \forall i \in V \setminus \{s, t\} && \text{(46c)}\\
0 \leq x_{ij} &\leq (1 - y_{ij})\, u_{ij} + (1 - y_{ji})\, u_{ji}, \forall (i,j) \in E && \text{(46d)}\\
y_{ij} &\in \{0, 1\}, \forall (i,j) \in E && \text{(46e)}
\end{aligned}$$

The objective (46) minimizes the path length and the negative of the value of the dynamic flow with path reversal. Equations (46a) are the constraints to save a path. Constraints (46b) ensure that the path chosen contains only the arcs with positive capacity. Constraints (46c) are flow-conservation constraints, and constraints (46d) limit the static flow rate on an arc (i, j) by u_{ij} if (j, i) is in the saved path, by u_{ji} if (i, j) is in the saved path, and by $u_{ij} + u_{ji}$ otherwise. For brevity, we refer to the model as BPMDC.

6.3.3 Solution strategy

To solve the problem, we apply the idea given in Theorem 6.9 to convert the problem into an ϵ-constrained mixed binary integer linear program which gives a weakly Pareto optimal (weakly efficient) solution for each $\epsilon \in \mathbb{R}$ and then develop a procedure that gives Pareto optimal solutions whenever the transit time function is positive integer-valued. We formulate the ϵ-constrained problem ($\epsilon \in \mathbb{R}$) as:

$$\min \quad -\theta \left[\sum_{j \in V_s^+} x_{sj} - \sum_{j \in V_s^-} x_{js} \right] + \sum_{(i,j) \in E} \tau_{ij} x_{ij} \tag{47}$$

subject to:

$$\sum_{j \in V_i^+} y_{ij} - \sum_{j \in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s, d\} \\ 1 & \text{if } i = d \end{cases} \tag{47a}$$

$$y_{ij} \leq u_{ij}, \forall (i,j) \in E \tag{47b}$$

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = 0, \forall i \in V \setminus \{s, t\} \tag{47c}$$

$$0 \leq x_{ij} \leq (1 - y_{ij})\, u_{ij} + (1 - y_{ji})\, u_{ji}, \forall (i,j) \in E \tag{47d}$$

$$y_{ij} \in \{0, 1\}, , \forall (i,j) \in E \tag{47e}$$

$$\sum_{(i,j) \in E} \tau_{ij} y_{ij} \leq \epsilon. \tag{47f}$$

In Algorithm 20, we propose a procedure which not only lists out the paths corresponding to the non-dominated values of (ψ_1, ψ_2) but also gives the corresponding static flow that can be temporally repeated to get the corresponding maximum dynamic flow.

Algorithm 20: Non-dominated solutions of BPMDC

```
Input  : $N = (V, E, u, \tau, s, t)$ with a depot $d$ such that $u_{ij} \in \mathbb{Z}_{\geq 0}, \tau_{ij} \in \mathbb{Z}_{>0}$
Output: Set of non-dominated saved paths with the maximum dynamic contraflow
1  $\epsilon_{\min}$ = length of a shortest $d$–$s$ path, $\epsilon_{\max} = \sum_{(i,j) \in E} \tau_{ij}$
2  LIST = { }
3  $\psi_2^0 = -\infty$
4  $\epsilon_1 = \epsilon_{\max} + 1$
5  $k = 1$
6  while $\epsilon_k > \epsilon_{\min}$ do
7      $Z_k$ = solution of problem (47) for $\epsilon = \epsilon_k - 1$
8      add $Z_k$ to LIST
9      $\psi_1^k = \psi_1(Z_k)$, $\psi_2^k = \psi_2(Z_k)$
10     if $\psi_2^k = \psi_2^{k-1}$ then
11         remove $Z_{k-1}$ from LIST
12     end
13     $\epsilon_{k+1} = \psi_1^k$
14     $k = k + 1$
15 end
16 For each $Z = \{(x_{ij})_{(i,j) \in E}, (y_{ij})_{(i,j) \in E}\} \in LIST$, construct the path using $(i, j)$
     with $y_{ij} = 1$ and the dynamic flow by temporal repetion of $x$.
```

Theorem 6.10. *Given $\tau_{ij} \in \mathbb{Z}_{>0}$, let $\mathcal{X}$ be the feasible set of BPMDC (46). Let $\mathcal{Y}_D$ be the set of all non-dominated points in $\psi(\mathcal{X})$, and $\mathcal{S} = \{(\psi_1(Z), \psi_2(Z)) : Z \in LIST\}$, then $\mathcal{S} = \mathcal{Y}_D$.*

Proof. Because of Theorem 6.9, $(\psi_1(Z_k), \psi_2(Z_k))$ is weakly non-dominated in $\psi(\mathcal{X})$ for each Z_k given in Line 7 of Algorithm 20. Because ϵ decreases by at least 1 in each iteration, the sequence $\{\psi_1(Z_k)\}$ is a strictly decreasing sequence. So because of Lines 10 and 11, $(\psi_1(Z_k), \psi_2(Z_k)) \in \mathcal{Y}_D$.

Conversely, let $Z^* \in \mathcal{X}$ such that $(\psi_1(Z^*), \psi_2(Z^*)) \in \mathcal{Y}_D$ and $(\psi_1(Z^*), \psi_2(Z^*)) \notin \mathcal{S}$. Clearly, the value of ψ_1 (a d–s path length) is bounded above by $\sum_{(i,j)\in E} \tau_{ij}$ and below by the length of the shortest path. Thus, if $LIST = \{Z_1, \cdots, Z_p\}$, there exists l such that $\psi_1(Z_l) > \psi_1(Z^*) > \psi_1(Z_{l+1})$ and $\psi_2(Z_l) < \psi_2(Z^*) < \psi_2(Z_{l+1})$ where $1 \leq l < p$. This implies that there eixsts a d–s path of length $\psi_1(Z^*)$ in N saving which gives a maximum dynamic contraflow value $-\psi_2(Z^*)$. So, putting $\epsilon = \psi_1(Z^*)$ in problem (47) gives the minimum value of ψ_2 as $\psi_2(Z^*)$. This is a contradiction, because for such an ϵ the minimum value of ψ_2 is $\psi_2(Z_{l+1})$ according to Algorithm 20 □

Although integrality restrictions are not put for the variable x in problem (46), if u, τ are integral, one can guarantee the existence of solutions in which x is integral.

Theorem 6.11. *Given $u_{ij} \in \mathbb{Z}_{\geq 0}, \tau_{ij} \in \mathbb{Z}_{>0}$, there exist solutions to problem (46) with x integral.*

Proof. Fixing y_{ij} to satisfy constraints (46a), (46e), the problem is equivalent to a maximum dynamic flow problem in the auxiliary network resulting from the removal of a d–s path. Since a maximum dynamic flow problem with integer inputs always has an integral solution, the result follows. □

The constraints (46a), (46e) are satisfied by a d–s path which may include cycles (subtours). However, the paths in the solutions generated by Algorithm 20 do not contain such cycles. Let us consider a path with a cycle. Removing the cycle decreases the value of ψ_1. However, it does not decrease the corresponding maximum dynamic contraflow value because removal of the cycle frees capacities which can be used by the flow. This leads to:

Theorem 6.12. *The saved paths in the solution of problem (46) are simple.*

6.3.4 Case illustration

For a case illustration, we consider Kathmandu road network inside the Ring Road considering only the major road-segments (no. of nodes $n = 44$, no. of arcs $m = 132$).

It is the same network that we have considered in Section 6.2.3 where we have taken node 24 as d.

Considering the time horizon of θ = 120 minutes, the non-dominated saved paths, their lengths, and the corresponding maximum dynamic contraflow value are shown in the following table.

Non-dominated path	Path length (mins.)	Flow value
24–25–26–35–15–14–13–12–0	27	71400
24–25–26–21–20–19–18–28–27–1–0	26	70320
24–25–26–21–20–19–18–28–27–0	19	70200
24–25–26–21–20–19–18–0	13	69960

Using the programming language Python 3.7 on a computer with Mac operating system having 1.8 GHz dual-core Intel Core i5 processor and 8 GB RAM, the solutions are obtained in less than a second. The solver used to solve the mixed integer program is CBC (Coin-OR branch and cut).

6.3.5 Bicriteria path-saving model with quickest contraflow

Assuming that

$$\psi_1 = \sum_{(i,j)\in E} \tau_{ij} y_{ij}$$

and

$$\psi_2 = \frac{Q + \sum_{(i,j)\in E} \tau_{ij} x_{ij}}{v},$$

the problem of minimizing the path length and the quickest time of the the flow of value Q can be formulated as:

$$\min \ (\psi_1, \psi_2) \tag{48}$$

subject to:

$$\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s, d\} \\ 1 & \text{if } i = d \end{cases} \tag{48a}$$

$$y_{ij} \leq u_{ij}, \forall (i,j) \in E \tag{48b}$$

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s, t\} \\ -v & \text{if } i = t \end{cases} \tag{48c}$$

$$0 \leq x_{ij} \leq (1-y_{ij})\,u_{ij} + (1-y_{ji})\,u_{ji}, \forall (i,j) \in E \tag{48d}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in E. \tag{48e}$$

The objective (48), minimizes the path length and the time horizon. Constraints (48a) are the constraints for the saved path. Constraints (48b) ensure that the path chosen contains only the arcs with positive capacity. Constraints (48c) are mass-balance constraints, and constraints (48d) limit the static flow rate on an arc (i,j) by u_{ij} if (j,i) is in the saved path, by u_{ji} if (i,j) is in the saved path, and by $u_{ij}+u_{ji}$ otherwise. For brevity, we refer to the model as BPQC.

Using the idea described to formulate problem (43), we can write the ϵ-constrained problem associated with (48) as follows.

$$\min \ \psi_2 = Q\omega + \sum_{(i,j)\in E} \tau_{ij}\xi_{ij} \tag{49}$$

subject to:

$$\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,d\} \\ 1 & \text{if } i = d \end{cases} \tag{49a}$$

$$y_{ij} \leq u_{ij}, \forall (i,j) \in E \tag{49b}$$

$$\sum_{j\in V_i^+} \xi_{ij} - \sum_{j\in V_i^-} \xi_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \in V \setminus \{s,t\} \\ -1 & \text{if } i = t \end{cases} \tag{49c}$$

$$0 \leq \xi_{ij} \leq \zeta_{ij}u_{ij} + \zeta_{ji}u_{ji}, \forall (i,j) \in E \tag{49d}$$

$$0 \leq \zeta_{ij} \leq \omega, \forall (i,j) \in E \tag{49e}$$

$$\omega - y_{ij} \leq \zeta_{ij} \leq 1 - y_{ij}, \forall (i,j) \in E \tag{49f}$$

$$\sum_{(i,j)\in E} \tau_{ij}y_{ij} \leq \epsilon \tag{49g}$$

$$y_{ij} \in \{0,1\}, \ \forall (i,j) \in E \tag{49h}$$

We can use the procedure used in Algorithm 20, replacing $-\infty$ in Line 3 by ∞, and problem (47) in Line 7 by problem (49), to find the non-dominated solutions of BPQC problem (48).

6.4 A Bilevel Programming Approach

6.4.1 Bilevel programming

A bilevel programming problem is an optimization problem in which one optimization problem (called the lower level problem) is among the constraints of the other (called the upper level problem). Partitioning the decision variables into two vectors x and y, let the lower level problem be:

$$\min_{x}\{f(x,y) : g(x,y) \leq 0, h(x,y) = 0\} \tag{50}$$

where

$$f : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}, g : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}^p, h : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}^q,$$

$$g(x,y) = (g_1(x,y), \cdots g_p(x,y))^\intercal, h(x,y) = (h_1(x,y), \cdots, h_q(x,y))^\intercal.$$

Let $\Psi : \mathbb{R}^{m_2} \to 2^{\mathbb{R}^{m_1}}$ be a set-valued mapping such that $\Psi(y)$ denotes the solution set of problem (50) for a fixed $y \in \mathbb{R}^{m_2}$, then the upper level problem is:

$$\min_{x,y}\{F(x,y) : G(x,y) \leq 0, H(x,y) = 0, x \in \Psi(y)\} \tag{50a}$$

where

$$F : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}, G : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}^k, H : \mathbb{R}^{m_1 \times m_2} \to \mathbb{R}^l,$$

$$G(x,y) = (G_1(x,y), \cdots G_k(x,y))^\intercal, H(x,y) = (H_1(x,y), \cdots, H_l(x,y))^\intercal.$$

For details, we refer to Dempe (2002).

In game theoretic terms, a bilevel programming problem may be considered as a Stackelberg leader-follower game, where the leader (representing upper level) makes the first move by choosing y, and the follower (representing the lower level) reacts by choosing x optimally. The leader selects, finally, such a y that together with x returned by the follower, the upper level objective $F(x,y)$ is optimized.

6.4.2 A bilevel path-saving model maximizing dynamic contraflow

Now, we formulate the problem of saving a path from a given node d to the source node s maximizing the dynamic flow allowing arc reversals to fulfill a given objective based on the path selection and resulting flow, as a bilevel programming problem. The upper level selects a d–s path to be saved and lower level maximizes the dynamic flow value on the resulting network allowing arc reversal (Nath, Pyakurel, Dempe, & Dhamala, 2019a).

The upper level problem is:

$$\min \quad F(x,y) \tag{51}$$

subject to:

$$\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s \\ 0 & \text{if } i \in V\setminus\{s,d\} \\ 1 & \text{if } i = d \end{cases} \tag{51a}$$

$$y_{ij} \le u_{ij}, \forall (i,j)\in E \tag{51b}$$

$$y_{ij} \in \{0,1\}, \forall (i,j)\in E \tag{51c}$$

where $x = (x_{ij})_{(i,j)\in E}, y = (y_{ij})_{(i,j)\in E}$, and x is obtained by solving the lower level problem:

$$\min \quad -\theta\left(\sum_{j\in V_s^+} x_{sj} - \sum_{j\in V_s^-} x_{js}\right) + \sum_{(i,j)\in E} \tau_{ij}x_{ij} \tag{51d}$$

subject to:

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = 0, \forall i \in V\setminus\{s,t\} \tag{51e}$$

$$0 \le x_{ij} \le (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \forall (i,j)\in E \tag{51f}$$

The constraints (51a)–(51c) construct a d–s path with $y_{ij} = 1$ if (i,j) lies on the path. The Constraints (51b) ensure that arcs with positive capacity can only be a part of the path. The lower level problem is a maximum dynamic contraflow problem. The Constraints (51f) bound the flow on an arc (i,j) by $u_{ij} + u_{ji}$ if neither (i,j) nor (i,i) is chosen for the path construction by the upper level. If (i,j) is chosen and (j,i) is not chosen for path construction, flow on (i,j) is bounded by u_{ji}.

The constraints (51a)-(51c) construct a d–s path which may also contain subtours. Unlike in the case of the shortest path problem with non-negative arc costs (lengths), where a solution without subtours is always better than the corresponding solution with subtours, a solution with a subtour may be better than than the one without a subtour in our consideration of the problem. Example 6.5 illustrates this fact.

Example 6.5. Suppose that the upper level objective function is

$$F(x,y) = \sum_{(i,j)\in E: u_{ij}\neq 0} w_{ij}(y_{ij} - x_{ij}/u_{ij}) \tag{52}$$

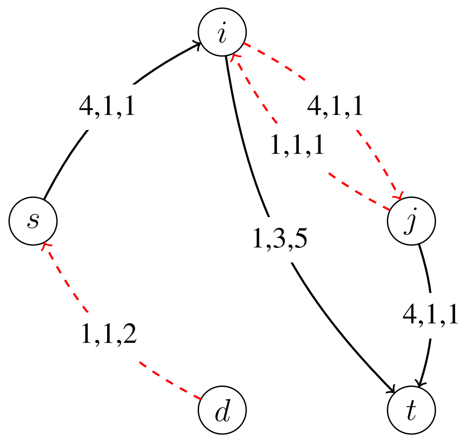


Figure 37: Path d–s along with a subtour i-j-i (the arc labels represent capacity, time, weight)

where the weight $w_{ij} \geq 0$ is the utility assigned to (i, j), as realized by the upper level, so as to motivate flow along the arc.

Consider a network shown in Figure 37. The only path from d to s is d–s. Given a time horizon $\theta = 5$, without taking any subtour, the static flow which generates the temporally repeated maximum dynamic s–t flow (allowing arc reversal) is given by: $x_{si} = x_{ij} = x_{jt} = 4, x_{it} = x_{ji} = x_{ds} = 0$ and the variables related to the path are $y_{ds} = 1, y_{si} = y_{ij} = y_{jt} = y_{it} = y_{ij} = 0$. The value of the objective (52) in this case is:

$$F_1 = \sum_{ij \in E : u_{ij} \neq 0} w_{ij} y_{ij} - \sum_{(i,j) \in E : u_{ij} \neq 0} \frac{w_{ij} x_{ij}}{u_{ij}} = 2 - 3 = -1$$

If the subtour i–j–i is considered along with the path, $x_{si} = x_{it} = 1$ and $y_{ds} = y_{ij} = y_{ji} = 1$ with the objective function value

$$F_2 = \sum_{(i,j) \in E : u_{ij} \neq 0} w_{ij} y_{ij} - \sum_{(i,j) \in E : u_{ij} \neq 0} \frac{w_{ij} x_{ij}}{u_{ij}} = 4 - (5 + 1/4) = -5/4 < F_1.$$

However, there may be some objectives which can be improved by subtour elimination, e.g. if the second part of the sum in the objective (52) involves the objective of the lower level problem, the subtours may be removed automatically because elimination of a subtour from the path chosen by the upper level, increases the capacity of the arcs for the flow on the lower level.

If subtour elimination is necessary, we can ensure the elementarity of the path (avoid subtours) including the following constraints in the upper level (Drexl & Irnich, 2014;

Bui, Deville, & Pham, 2016; Taccari, 2016).

$$\begin{aligned}
\sum_{(i,j)\in E(S)} y_{ij} &\leq \sum_{i\in S\setminus\{k\}} \sum_{j\in V_i^+} y_{ij} \; \forall k \in S, \forall S \subseteq V \setminus \{s,d\}, |S| \geq 2 && \text{(53a)} \\
y_{id} &= 0, \forall i \in V_d^- && \text{(53b)} \\
y_{sj} &= 0, \forall j \in V_s^+ && \text{(53c)}
\end{aligned}$$

where $E(S)$ denotes the set of arcs with both the ends in S. Known as generalized cut-set (GCS) inequalities, these inequalities increase the number of constraints by $O(n2^n)$. Other techniques with polynomial number of additional variables and additional constraints can also be found in Taccari (2016). For example, adding (53b), (53c) and the following constraints

$$\pi_i - \pi_j + ny_{ij} \leq n - 1, \forall (i,j) \in E \tag{54}$$

in the upper level (see also Bui et al. (2016)), the number of additional variables $\pi_i (i = 1, \cdots, n)$ is $O(n)$, and that of additional constraints is $O(m)$. However, it is shown in the literature that the computational performance of GCS inequalities is better than other formulations in solving the shortest elementary path problem.

6.4.3 Solution strategies

We present two approaches to solve the problem. One uses the idea of Stackelberg game along with the maximum dynamic contraflow algorithm, particularly useful when one can get all the paths form d to s in a desired time, and the other converts the bilevel program to a single-level mixed binary integer linear program so that one can use the algorithms to solve mixed-integer programming problems.

A Naïve Algorithm: A straight-forward procedure to solve the problem is presented in Algorithm 21. The algorithm iteratively chooses a d–s path and uses maximum dynamic contraflow algorithm (see Section 3.2.1) in the network formed by excluding the path and calculates the objective function value. The output is the path which gives the best objective function value, along with the related flow.

KKT approach: A common approach to solve a bilevel programming problem is to transform it into a single level optimization problem replacing the lower level problem by its Karush-Kuhn-Tucker (KKT) conditions. This results into what is known as a mathematical program with equilibrium or complementarity constraints (MPEC or MPCC) (see Luo, Pang, and Ralph (1996)).

Algorithm 21: Naïve algorithm

```
Input : Directed network $N = (V, E, u, \tau, s, t)$ with a depot node $d$, and a time
        horizon $\theta$
1  $Val_{OBJ} = \infty$
2  LIST = list of simple paths from $d$ to $s$ without zero capacity arcs
3  for $P$ in LIST do
4      Construct the auxiliary network $N'$ excluding the arcs in $P$
5      Calculate the static flow $x$ corresponding to the maximum dynamic flow on $N'$
6      $\text{Val}_{OBJ}^{new} =$ value of the objective function with $y$-values corresponding to $P$ and
        $x$ from Line 5
7      If $\text{Val}_{OBJ}^{new} < \text{Val}_{OBJ}$, then $P^* = P, x^* = x$
8      Retain the network
9  end
10 Return $P^*, x^*$
```

Consider the Lagrangian

$$\begin{aligned} Lg(x, y, \lambda, \mu) &= -\theta\left(\sum_{j\in V_s^+} x_{sj} - \sum_{j\in V_s^-} x_{js}\right) + \sum_{(i,j)\in E} \tau_{ij}x_{ij} \\ &\quad + \sum_{i\in V\setminus\{s,d\}} \lambda_i \left(\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji}\right) \\ &\quad + \sum_{(i,j)\in E} \mu_{ij}\left[x_{ij} - (1 - y_{ij})u_{ij} - (1 - y_{ji})u_{ji}\right] \end{aligned}$$

The KKT conditions for the lower label problem are (51e), (51f) along with

$$\begin{aligned} \lambda_i - \lambda_j + \mu_{ij} &\geq -\tau_{ij},\ \forall (i,j) \in E && (55a)\\ \mu_{ij}\left[x_{ij} - (1 - y_{ij})u_{ij} - (1 - y_{ji})u_{ji})\right] &= 0,\ \forall (i,j) \in E && (55b)\\ \mu_{ij} &\geq 0,\ \forall (i,j) \in E && (55c)\\ \lambda_i &\in \mathbb{R},\ \forall i \in V \setminus \{s,t\} && (55d) \end{aligned}$$

Although there is no λ associated with s and t, the inequalities (55a) are valid if we put $\lambda_s = -\theta$ and $\lambda_t = 0$.

In what follows, we write the upper level constraints as ULC. Thus the single-level problem corresponding to the bilevel problem is the problem (51) subject to ULC, (51e), (51f), (55a)–(55d), which is a mixed integer non-linear optimization problem.

The nonlinearity is because of the constraints (55b), and solution of this problem would give local optimal solutions. However, since the Mangasarian-Fromovitz constraint qualifications are violated at every feasible point of this problem, the nonlinear optimization solvers may fail to obtain a local optimal solution. This can be overcome by solving the following relaxation of the problem for $\epsilon \downarrow 0$ (Dempe, 2019).

$$\min \; F(x,y) \tag{56}$$

subject to

$$\begin{aligned}
&\text{ULC} && (56a)\\
\sum_{j\in A_i^+} x_{ij} - \sum_{j\in A_i^-} x_{ji} &= 0, \forall i \in V \setminus \{s,t\} && (56b)\\
0 \le x_{ij} &\le (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \forall (i,j) \in E && (56c)\\
\lambda_i - \lambda_j + \mu_{ij} &\ge -\tau_{ij},\ \forall (i,j) \in E, i \ne s, j \ne t && (56d)\\
-\lambda_j + \mu_{sj} &\ge \theta - \tau_{sj},\ \forall j \in V_s^+ && (56e)\\
\lambda_i + \mu_{ij} &\ge -\tau_{ij},\ \forall i \in V_t^-, && (56f)\\
\mu_{ij} &\ge 0,\ \forall (i,j) \in E && (56g)\\
\epsilon &\ge \mu_{ij}\left[x_{ij} - (1-y_{ij})u_{ij} - (1-y_{ji})u_{ji}\right],\ \forall (i,j) \in E && (56h)
\end{aligned}$$

Solving problem (56), we get the local optimal solutions. But in our consideration of the problem, a global solution is desirable. To get a global optimal solution, we can employ, what is popularly known as, a big M method. Choosing large enough M (Fortuny-Amat & McCarl, 1981; Pineda, Bylling, & Morales, 2018), we can replace (55b) by

$$\begin{aligned}
-x_{ij} + (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji} &\le M' z_{ij}, \forall (i,j) \in E && (57a)\\
\mu_{ij} &\le (1-z_{ij})M'', \forall (i,j) \in E && (57b)
\end{aligned}$$

Let $U = \max\{u_{ij} : (i,j) \in E\}$. Then the maximum possible value of $(1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}$ is $2U$. Since the minimum possible value of x_{ij} is zero, we can safely take M' as $2U$. In this way, we can get the global optimal solution of the problem by solving the following formulation (58), with large enough M.

$$\min \; F(x,y) \tag{58}$$

subject to

$$\text{ULC} \tag{58a}$$

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = 0, \forall i \in V \setminus \{s,t\} \tag{58b}$$

$$0 \le x_{ij} \le (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \forall (i,j) \in E \tag{58c}$$

$$\lambda_i - \lambda_j + \mu_{ij} \ge -\tau_{ij},\ \forall (i,j) \in E, i \neq s, j \neq t \tag{58d}$$

$$-\lambda_j + \mu_{sj} \ge \theta - \tau_{sj},\ \forall j \in V_s^+ \tag{58e}$$

$$\lambda_i + \mu_{id} \ge -\tau_{id},\ \forall i \in V_t^-, \tag{58f}$$

$$2Uz_{ij} \ge -x_{ij} + (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \forall (i,j) \in E \tag{58g}$$

$$\mu_{ij} \le (1-z_{ij})M, \forall (i,j) \in E \tag{58h}$$

$$\mu_{ij} \ge 0,\ \forall (i,j) \in E \tag{58i}$$

$$z_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{58j}$$

But caution has to be taken choosing a big M when we use a solver software to solve the problem, because very low or very high values of M lead to infeasible, suboptimal, and numerically unstable solutions.

A method described in Pineda et al. (2018), to solve linear bilevel optimization problems, may be adapted to tune the values of big M. The idea is to use the local optimal solutions of the relaxed problem (56) to adjust the value of M. The procedure is presented in Algorithm 22.

Algorithm 22: Tuning big M

```
1 Initialize $\epsilon > 0, \delta > 1, H > 1$, and the number of iterations $K \in \mathbb{N}$.
2 Set initial number of iterations $k = 0$.
3 Solve problem (56) without (56h). Let the value of the upper level variable
    $y = (y_e)_{e \in A}$ be $y_0$ and the value of the dual variable $\mu = (\mu_e)_{e \in A}$ be $\mu_0$.
4 Find the optimal flow $x = x_0$ using the solution $y = y_0$ of the upper level.
5 while $k < K$ do
6     Increase $k$ by 1. Solve problem (56) by taking $x = x_{k-1}, y = y_{k-1}, \mu = \mu_{k-1}$ as
        an initial solution. Repace $\epsilon$ by $\epsilon/\delta$.
7 end
8 $M = H \times \max\{(\mu_k)_{ij} : (i,j) \in E\}$.
9 Return $M$.
```

The big M in (58h) is replaced by the value returned by Algorithm 22, and a mixed integer programming solver is used to solve problem (58). We may take $z_{ij} = 0$ if $-x_{ij} + (1-y_{ij})u_{ij} + (1-y_{ji}) > 0$ and $z_{ij} = 1$ if $\mu_{ij} > 0$ to initialize the variable z.

6.4.4 Identification of reversed arcs

From the solution of the aforementioned problem, we can also determine the arcs which are to be reversed. This can be done with the help of the optimal flow x^*. As we know that x^* may violate feasibility with respect to the capacities of the network N, i.e. x^*_{ij} may exceed u_{ij} for some (i, j). This implies that the opposite arc (j, i) has to be reversed to increase the capacity in the direction of (i, j). The procedure is mentioned in Algorithm 23.

Algorithm 23: Identifying reversed arcs

```
1 Identify the optimal path $P^*$ and the static flow $x^*$ corresponding to the dynamic
    flow
2 $R = \{(j, i) \in E \setminus P : x^*_{ij} > u_{ij}\}$
3 return R
```

Because of our assumption that $\tau_{ij} > 0$ for each $(i, j) \in E$, there will be no positive flow in cycles in the flow decomposition of x^* according to Corollary 6.2 . So, we do not require to decompose x^* into paths and cycles in Algorithm 23.

CHAPTER 7

SUMMARY AND CONCLUSIONS

In this chapter, we summarize the results, mention the limitations of the study, and conclude the work with suggesting some further research directions.

7.1 Summary of the Results

Because of the large size of the network to be dealt with in evacuation optimization, faster algorithms are always desirable. Improving the existing algorithm (requiring several calls of a minimum cost flow algorithm) to solve the quickest flow problem, we propose an algorithm that runs within a strongly polynomial time complexity of a minimum cost flow problem (Section 3.2.2). Computational results in Section 3.2.3 show that the decrease in the quickest time increases with the increase in the number of evacuee-cars. When the number of cars reaches 50000, the evacuation time is found to decrease by 42% because of contraflow configuration.

Realizing the need of reversing the direction of the traffic flow in the road segments up to the necessary capacity only, we introduce the partial contraflow approach (Section 3.3) and propose algorithms to solve the maximum static partial contraflow problem, maximum dynamic partial contraflow problem, quickest partial contraflow problem. All the algorithms run in strongly polynomial time.

When the direction of the traffic flow in a road segment is reversed, the travel time of the corresponding arc may be different than that of the original orientation, we propose a method of constructing an auxiliary network in Section 3.4 so that the procedures in the corresponding algorithms can be adapted to solve the problems in this new setting.

In contrast to the networks defined to be consisting of nodes and arcs with some attributes, abstract networks (which generalize the concept of networks) are defined to consist of elements (arcs or nodes) and paths (an ordered set of elements). We extend the concept of the partial contraflow approach in the abstract networks in Section 3.5 to construct polynomial-time algorithms to solve abstract maximum partial static partial contraflow and abstract maximum dynamic partial contraflow problems.

Using the bow network construction, available in the literature, to solve the quickest flow problem with inflow-dependent transit times approximately, we design a strongly polynomial $(2 + \epsilon)$-approximation algorithm to solve the corresponding contraflow/ partial contraflow problem in Section 4.3. Using BPR function and Davidson's function as inflow-dependent transit time functions, we test the performance of the algorithm in a Kathmandu road network. The ratio of the approximate quickest time to the optimal quickest time is found to be at most 1.098. It is also observed that if the average of the inflow-dependent transit time on an arc is considered to be the constant transit time, the quickest time with inflow-dependent transit times is almost equal to the quickest time with constant transit times. In Section 4.5, we construct corresponding algorithm for the case of density-dependent transit times.

In cases when a given set of facilities are to be assigned to a given set of arcs so that the capacities of the corresponding arcs are reduced resulting in an increase in the quickest time, we introduce the quickest FlowLoc problem in Section 5.3. We show that the single facility case of the problem can be solved in strongly polynomial time with the help of an algorithm that iterates over all the arcs in the set of arcs to which the facilities can be assigned. Proving that the multi-facility case is NP-hard, we propose two heuristics whose performance is tested in Section 5.3.3 considering a case of the Kathmandu road network. The faster heuristic has an average optimality gap of 3.48% and an average running time of 0.17 seconds. The corresponding values for the slower heuristic are 0.18% and 1.02 seconds. Algorithms to solve the quickest FlowLoc problem with a possibility of arc reversals are designed in Section 5.4.

To choose a single shelter location, when multiple choices for the shelters are available, we introduce the sink location problem as MaxStatic, MaxDynamic, Quickest sink location problems depending on the objective of maximizing the static flow, dynamic flow or minimizing the evacuation time in Section 5.5. We prove that such a problem can be solved in strongly polynomial time and suggest algorithms for the corresponding type of shelter location with arc reversals also.

By reversing the direction of the traffic flow towards the sink, a contraflow configuration may obstruct the paths towards the source. This hampers the movement of facilities, if required, towards the source. In Chapter 6, we introduce a path-saving approach. Saving a path not exceeding a given time of travel, we model the problem of maximizing the dynamic contraflow as a mixed binary integer linear programming problem. The analogous problem of minimizing the evacuation time is a mixed binary integer programming problem with a fractional objective. We suggest a linearization strategy so that the algorithms to solve the mixed-integer linear programming problems can be used. The solution, using available software solvers, considering a road network of Kathmandu city (consisting of 44 nodes and 132 arcs) can be obtained within 0.1 seconds.

The problem of minimizing the path length and maximizing the dynamic contraflow has been modeled as a bicriteria optimization problem (Section 6.3). A procedure using ϵ-constrained method is constructed to obtain efficient solutions. The computation considering the above-mentioned Kathmandu road network takes less than a second to obtain efficient solutions. We also model the problem of minimizing the path length and evacuation time as a bicriteria problem and construct a procedure to solve it.

To choose a path to optimize a general objective, maximizing the dynamic contraflow is modeled as a bilevel optimization problem (Section 6.4), in which the upper level chooses a path and the lower level maximizes the dynamic contraflow depending on the path-choice of the upper level. We use KKT approach to solve convert the problem into a single level problem which is a mixed binary integer non-linear programming problem. We describe linearizing strategy using a big M method with a procedure to tune the big M to an appropriate value.

7.2 Conclusions

In this work, we have developed some models and algorithms to maximize the flow (number of evacuees) or to minimize the time horizon of the flow (evacuation time horizon). We have tested the computational performance of the designed algorithms in most of the cases. Except for the path-saving models, the presented algorithms are theoretically and practically efficient. The solution procedures in the path saving models are also practically efficient in the network considered. The performance of the solution procedure for the proposed bilevel path-saving model is yet to be tested.

The overall contribution of the thesis can be listed as follows.

1. To record unused capacities, the partial contraflow approach has been introduced to reverse the direction of the traffic up to the necessary capacity.
2. Extension of contraflow/partial contraflow approach to networks with orientation-dependent, inflow-dependent, and density-dependent transit times on arcs has been done.
3. To find the optimal allocation of facilities to the arcs to find the minimum time horizon of the evacuation of a given number of evacuees, the quickest FlowLoc model has been introduced in a single-source-single-sink network. The solution

procedures with and without the contraflow approach have been designed.

4. Modeling of the problems to find the sink location from among a given set of possible sink locations with an objective of maximizing the flow or minimizing the time has been done. The complexity analysis of such problems with and without contraflow has been done.

5. Since, the reversals of arcs towards the shelter in the contraflow approach may obstruct paths towards the source, models to save a path to move facilities towards the source have been introduced. A bicriteria approach and a bilevel approach have also been used in the modeling.

6. The performance of most of the algorithms proposed has been tested by computer programming implementations considering a part of the Kathmandu road network as input.

However, the models have to be used keeping in mind the following limitations.

1. Except for the static flow cases, the dynamic flow cases are considered to be with a single source and single sink.

2. In all the models considered, the units of flow have identical attributes.

3. The cost of management of traffic and the other management issues during an evacuation are not incorporated in the models.

4. When applying contraflow approach, the lane reversal is done at the beginning of evacuation and the direction of the traffic flow remains intact until the evacuation is over.

5. The time required to reverse a lane is not considered.

7.3 Recommendations for Further Work

We have considered problems with arc reversals with the objective of maximizing the flow value or minimizing the evacuation time. These problems can have multiple solutions with the same objective value. This leads to the problem of finding the maximum flow or minimum evacuation time by reversing the minimum number of arcs or saving the maximum capacity.

To solve the single facility quickest FlowLoc problems or the sink location problems, we have proposed algorithms that, at least in worst cases, iterate over all the feasible locations. This proves that the corresponding problems can be solved in strongly polynomial times. Moreover, the iterative algorithms can also be used to solve the problems efficiently. However, it is still desirable to find algorithms with better running time. Since our heuristic to solve the multi-facility quickest FlowLoc problem with a better optimality gap solves the single facility case iteratively, the improvement of the running time of the single facility case naturally improves the running time of the heuristic. Moreover, the heuristics use some kind of greedy approach, there is room for a better heuristic in terms of optimality gap. Further, for quickest FlowLoc we assume in Section 5.2 that the number of available facilities does not exceed the total number of facilities that have to be placed on the set L and the size of the facility does not exceed the capacity of each arc in L, the corresponding problem lifting these restrictions is a natural generalization of the problem.

For the solutions of path-saving models described in Chapter 6, we rely on the algorithms of solving mixed-integer programming problems. The modifications of these algorithms using the special structure of the underlying graph can also be a direction of further research.

REFERENCES

Adhikari, I. M., & Dhamala, T. N. (2020). Minimum clearance time on the prioritized integrated evacuation network. *American Journal of Applied Mathematics*, *8*(4), 207–215.

Adhikari, I. M., Pyakurel, U., & Dhamala, T. N. (2020). An integrated solution approach for the time minimization evacuation planning problem. *International Journal of Operation Research*, *17*(1), 27–39.

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice hall.

Ahuja, R. K., & Orlin, J. B. (1989). A fast and simple algorithm for the maximum flow problem. *Operations Research*, *37*(5), 748–759.

Ahuja, R. K., & Orlin, J. B. (1991). Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems. *Naval Research Logistics (NRL)*, *38*(3), 413–430.

Ahuja, R. K., & Orlin, J. B. (1995). A capacity scaling algorithm for the constrained maximum flow problem. *Networks*, *25*(2), 89–98.

Akter, S., & Wamba, S. F. (2019). Big data and disaster management: a systematic review and agenda for future research. *Annals of Operations Research*, *283*(1-2), 939–959.

An, S., Cui, N., Li, X., & Ouyang, Y. (2013). Location planning for transit-based evacuation under the risk of service disruptions. *Transportation Research Part B: Methodological*, *54*, 1–16.

Bish, D. R. (2011). Planning for a bus-based evacuation. *OR spectrum*, *33*(3), 629–654.

Bui, Q. T., Deville, Y., & Pham, Q. D. (2016). Exact methods for solving the elementary shortest and longest path problems. *Annals of Operations Research*, *244*(2), 313–348.

Burkard, R. E., Dlaska, K., & Klinz, B. (1993). The quickest flow problem. *Zeitschrift für Operations Research*, *37*(1), 31–58.

Caunhye, A. M., Nie, X., & Pokharel, S. (2012). Optimization models in emergency logistics: A literature review. *Socio-economic planning sciences*, *46*(1), 4–13.

Cova, T. J., & Johnson, J. P. (2003). A network flow model for lane-based evacuation routing. *Transportation research part A: Policy and Practice*, *37*(7), 579–604.

Dantzig, G. B. (1998). *Linear programming and extensions* (Vol. 48). Princeton university press.

Dempe, S. (2002). *Foundations of bilevel programming*. Springer Science & Business Media.

Dempe, S. (2019). Computing locally optimal solutions of the bilevel optimization problem using the KKT approach. In *International conference on mathematical optimization theory and operations research* (pp. 147–157).

Dhamala, T. N., Adhikari, I. M., Nath, H. N., & Pyakurel, U. (2018). Meaningfulness of OR models and solution strategies for emergency planning. In *Living under the threat of earthquakes* (pp. 175–194). Springer.

Dhamala, T. N., & Pyakurel, U. (2013). Earliest arrival contraflow problem on series-parallel graphs. *International Journal of Operations Research*, *10*(1), 1–13.

Dhamala, T. N., Pyakurel, U., & Dempe, S. (2018). A critical survey on the network optimization algorithms for evacuation planning problems. *International Journal of Operations Research*, *15*(3), 101–133.

DHS. (2004). *National response plan*. U.S. Department of Homeland Security.

Dhungana, R. C., & Dhamala, T. N. (2019). Maximum FlowLoc problems with network reconfiguration. *International Journal of Operations Research*, *16*(1), 13–26.

Dhungana, R. C., & Dhamala, T. N. (2020). Flow improvement in evacuation planning with budget constrained switching costs. *International Journal of Mathematics and Mathematical Sciences*, *2020*.

Dhungana, R. C., Pyakurel, U., & Dhamala, T. N. (2018). Abstract contraflow models and solution procedures for evacuation planning. *Journal of Mathematics Research*, *10*(4), 89–100.

Drexl, M., & Irnich, S. (2014). Solving elementary shortest-path problems as mixed-integer programs. *OR spectrum*, *36*(2), 281–296.

Edmonds, J., & Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, *19*(2), 248–264.

Ehrgott, M. (2005). *Multicriteria optimization* (Vol. 491). Springer Science & Business Media.

Fleischer, L., & Tardos, É. (1998). Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, *23*(3), 71–80.

Ford, L. R., & Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, *6*, 419–433.

Ford, L. R., & Fulkerson, D. R. (1962). *Flows in networks*. New Jersey: Princeton University Press.

Fortuny-Amat, J., & McCarl, B. (1981). A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, *32*(9), 783–792.

Goerigk, M., Deghdak, K., & Heßler, P. (2014). A comprehensive evacuation planning model and genetic soution algorithm. *Transportation Research, Part E*, *71*, 82–97.

Goerigk, M., Grün, B., & Heßler, P. (2013). Branch and bound algorithms for the bus evacuation problem. *Computers & Operations Research*, *40*(12), 3010–3020.

Goerigk, M., Grün, B., & Heßler, P. (2014). Combining bus evacuation with location decisions: A branch-and-price approach. *Transportation Research Procedia*, *2*, 783–791.

Goldberg, A. V. (1985). *A new max-flow algorithm.* Laboratory for Computer Science, Massachusetts Institute of Technology.

Goldberg, A. V., & Tarjan, R. (1987). Solving minimum-cost flow problems by successive approximation. In *Proceedings of the nineteenth annual ACM symposium on theory of computing* (pp. 7–18).

Goldberg, A. V., & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, *35*(4), 921–940.

Goldberg, A. V., & Tarjan, R. E. (1989). Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)*, *36*(4), 873–886.

Hamacher, H. W., Heller, S., & Rupp, B. (2013). Flow location (FlowLoc) problems: dynamic network flow and location models for evacuation planning. *Annals of Operations Research*, *207*(1), 161–180.

Hamacher, H. W., & Tjandra, S. A. (2001). *Mathematical modelling of evacuation problems: A state of art.* Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer (ITWM).

Heller, S., & Hamacher, H. W. (2011). The multi terminal q-FlowLoc problem: A heuristic. In *International conference on network optimization* (pp. 523–528).

Hu, Y., Zhao, X., Liu, J., Liang, B., & Ma, C. (2020). An efficient algorithm for solving minimum cost flow problem with complementarity slack conditions. *Mathematical Problems in Engineering.* doi: https://doi.org/10.1155/2020/2439265

Jewell, W. (1958). Optimal flow through networks interim technical report 8. *Massachusetts Institute of Technology, Cambridge, MA.*

Jia, H., Ordóñez, F., & Dessouky, M. (2007). A modeling framework for facility location of medical services for large-scale emergencies. *IIE Transactions*, *39*(1), 41–55.

Kappmeier, J.-P. W. (2015). *Generalizations of flows over time with applications in evacuation optimization* (PhD thesis). TU Berlin.

Kappmeier, J.-P. W., Matuschke, J., & Peis, B. (2014). Abstract flows over time: A first step towards solving dynamic packing problems. *Theoretical Computer Science*, *544*, 74–83.

Karzanov, A. V. (1974). Determining the maximal flow in a network by the method of preflows. In *Soviet math. doklady* (Vol. 15, pp. 434–437).

Kim, S., Shekhar, S., & Min, M. (2008). Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, *20*(8), 1115–1129.

Klein, M. (1967). A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, *14*(3), 205–220.

Köhler, E., Langkau, K., & Skutella, M. (2002). Time-expanded graphs for flow-dependent transit times. In *European symposium on algorithms* (pp. 599–611).

Köhler, E., & Skutella, M. (2005). Flows over time with load-dependent transit times. *SIAM Journal on Optimization*, *15*(4), 1185–1202.

Kongsomsaksakul, S., Yang, C., & Chen, A. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, *6*, 4237–4252.

Kotsireas, I. S., Nagurney, A., & Pardalos, P. M. (2015). Dynamics of disasters-key concepts, models, algorithms, and insights. *Springer Proceedings in Mathematics and Statistics*.

Kulshrestha, A., Lou, Y., & Yin, Y. (2014). Pick-up locations and bus allocation for transit-based evacuation planning with demand uncertainty. *Journal of Advanced Transportation*, *48*(7), 721–733.

Langkau, K. (2003). *Flows over time with flow-dependent transit times* (PhD thesis). TU Berlin.

Lin, M., & Jaillet, P. (2015). On the quickest flow problem in dynamic networks: a parametric min-cost flow approach. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms* (pp. 1343–1356).

Luo, Z.-Q., Pang, J.-S., & Ralph, D. (1996). *Mathematical programs with equilibrium constraints*. Cambridge University Press.

Martens, M., & McCormick, S. T. (2008). A polynomial algorithm for weighted abstract flow. In *International conference on integer programming and combinatorial optimization* (pp. 97–111).

McCormick, S. T. (1996). A polynomial algorithm for abstract maximum flow. In *Proceedings of the 7th annual ACM-SIAM symposium on discrete algorithms* (pp. 490–497).

Minieka, E. (1973). Maximal, lexicographic, and dynamic network flows. *Operations Research*, *21*(2), 517–527.

Mtoi, E. T., & Moses, R. (2014). Calibration and evaluation of link congestion functions. *Journal of Transportation Technologies*, *4*(2), 141–149.

Nath, H. N., Dempe, S., & Dhamala, T. N. (2021). A bicriteria approach for saving path maximizing dynamic contraflow. *Asia-Pacific Journal of Operational Research*.

doi: https://doi.org/10.1142/S0217595921500275

Nath, H. N., & Dhamala, T. N. (2017). Identification of optimal pick-up locations with their demands in evacuation planning of transit-dependent population. In *Proceeding of national conference on mathematics and its applications (NCMA–21017)* (pp. 48–53).

Nath, H. N., & Dhamala, T. N. (2018). Network flow approach for locating optimal sink in evacuation planning. *International Journal of Operations Research*, *15*(4), 175–185.

Nath, H. N., Pyakurel, U., Dempe, S., & Dhamala, T. N. (2019a). A bilevel programming approach to save a path maximizing the dynamic flow with lane reversals for evacuation planning. Preprint, TU Bergakademie, Freiberg.

Nath, H. N., Pyakurel, U., Dempe, S., & Dhamala, T. N. (2019b). A path saving strategy with arc reversals for evacuation planning. *International Journal of Innovative Knowledge Concepts*, *7*(1), 160–165.

Nath, H. N., Pyakurel, U., & Dhamala, T. N. (2018). Facility location on arcs for quickest evacuation planning. In *The 11th triennial conference of association of Asia Pacific operational research societies, APORS–2018, abstract and program book* (pp. 115–117).

Nath, H. N., Pyakurel, U., & Dhamala, T. N. (2021). Network reconfiguration with orientation-dependent transit times. *International Journal of Mathematics and Mathematical Sciences*, *2021*. doi: 10.1155/2021/6613622

Nath, H. N., Pyakurel, U., Dhamala, T. N., & Dempe, S. (2021). Dynamic network flow location models and algorithms for quickest evacuation planning. *Journal of Industrial and Management Optimization*, *17*(5), 2925–2941.

Ng, M., Park, J., & Waller, S. T. (2010). A hybrid bilevel model for the optimal shelter assignment in emergency evacuations. *Computer-Aided Civil and Infrastructure Engineering*, *25*(8), 547–556.

Orlin, J. B. (1993). A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, *41*(2), 338–350.

Orlin, J. B. (2013). Max flows in $O(nm)$ time, or better. In *Proceedings of the forty-fifth annual acm symposium on theory of computing* (pp. 765–774).

Pineda, S., Bylling, H., & Morales, J. (2018). Efficiently solving linear bilevel programming problems using off-the-shelf optimization software. *Optimization and Engineering*, *19*(1), 187–211.

Pyakurel, U. (2016). *Evacuation planning problem with contraflow approach* (PhD thesis). IOST, Tribhuvan Univeristy, Kathmandu, Nepal.

Pyakurel, U., & Dhamala, T. N. (2015). Models and algorithms on contraflow evacuation planning network problems. *International Journal of Operations Research*, *12*(2), 36–46.

Pyakurel, U., & Dhamala, T. N. (2016). Continuous time dynamic contraflow models and algorithms. *Advances in Operations Research*, *2016*. doi: http://dx.doi.org/10.1155/2016/7902460

Pyakurel, U., & Dhamala, T. N. (2017a). Continuous dynamic contraflow approach for evacuation planning. *Annals of Operations Research*, *253*(1), 573–598.

Pyakurel, U., & Dhamala, T. N. (2017b). Evacuation planning by earliest arrival contraflow. *Journal of Industrial & Management Optimization*, *13*(1), 489–503.

Pyakurel, U., Dhamala, T. N., & Dempe, S. (2017). Efficient continuous contraflow algorithms for evacuation planning problems. *Annals of Operations Research*, *254*(1-2), 335–364.

Pyakurel, U., Goerigk, M., Dhamala, T. N., & Hamacher, H. W. (2015). Transit dependent evacuation planning for Kathmandu valley: a case study. *International Journal of Operations Research Nepal*, *5*, 49–73.

Pyakurel, U., Hamacher, H. W., & Dhamala, T. N. (2014). Generalized maximum dynamic contraflow on lossy network. *International Journal of Operations Research Nepal*, *3*(1), 27–44.

Pyakurel, U., Nath, H. N., Dempe, S., & Dhamala, T. N. (2019). Efficient dynamic flow algorithms for evacuation planning problems with partial lane reversal. *Mathematics*, *7*(10). doi: https://doi.org/10.3390/math7100993

Pyakurel, U., Nath, H. N., & Dhamala, T. N. (2018). Efficient contraflow algorithms for quickest evacuation planning. *Science China Mathematics*, *61*(11), 2079–2100.

Pyakurel, U., Nath, H. N., & Dhamala, T. N. (2019). Partial contraflow with path reversals for evacuation planning. *Annals of Operations Research*, *283*(1-2), 591–612.

Rebennack, S., Arulselvan, A., Elefteriadou, L., & Pardalos, P. M. (2010). Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, *19*(2), 200–216.

Röck, H. (1980). Scaling techniques for minimal cost network flows. *Discrete Structures and Algorithms*, 181–191.

Rupp, B. (2010). *Flowloc: Discrete facility locations in flow networks* (Diploma thesis). TU Kaiserslautern.

Saho, M., & Shigeno, M. (2017). Cancel-and-tighten algorithm for quickest flow problems. *Networks*, *69*(2), 179–188.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Englewood Cliffs, NJ.

Sherali, H. D., Carter, T. B., & Hobeika, A. G. (1991). A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B: Methodological*, *25*(6), 439–452.

Skutella, M. (2009). An introduction to network flows over time. In *Research trends in* *combinatorial optimization* (pp. 451–482). Springer.

Taccari, L. (2016). Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, *252*(1), 122–130.

Torres, F. E. (1990). Linearization of mixed-integer products. *Mathematical programming*, *49*(1), 427–428.

Vogiatzis, C., Walteros, J. L., & Pardalos, P. M. (2013). Evacuation through clustering techniques. In *Models, algorithms, and technologies for network analysis* (pp. 185–198). Springer.

APPENDIX A

NETWORK DATA FOR COMPUTATIONS

A.1 Virtual Network in Section 3.2.3

Node (i)	Nodes adjacent to i (capacity(per second), travel time (minutes))
$1(s)$	$2(2,9)$, $5(3,6)$
2	$1(3,9)$, $15(1,10)$, $3(1,5)$, $6(2,8)$
3	$2(1,5)$, $4(1,9)$, $7(1,8)$
4	$3(2,9)$, $8(3,9)$
5	$1(2,6)$, $6(3,8)$, $9(1,9)$
6	$10(2,7)$, $2(2,8)$, $5(1,8)$, $7(3,5)$
7	$11(3,9)$, $13(1,5)$, $18(3,5)$, $3(2,8)$, $6(1,5)$, $8(2,8)$
8	$4(2,9)$, $7(2,8)$, $12(2,9)$
9	$5(2,9)$, $13(1,6)$
10	$11(3,7)$, $14(3,7)$, $6(2,7)$
11	$10(2,7)$, $12(2,9)$, $15(1,5)$, $17(2,6)$, $21(1,10)$, $7(3,9)$
12	$11(3,9)$, $16(2,5)$, $23(2,8)$, $8(1,9)$
13	$14(2,6)$, $17(2,6)$, $7(2,5)$, $9(2,6)$
14	$10(3,7)$, $13(3,6)$, $15(1,9)$, $18(2,10)$, $22(1,10)$
15	$11(3,5)$, $14(1,9)$, $16(2,10)$, $19(3,5)$, $2(2,10)$, $24(1,10)$
16	$12(2,5)$, $15(2,10)$, $20(3,7)$
17	$11(3,6)$, $13(3,6)$, $18(2,9)$
18	$14(3,10)$, $17(1,9)$, $20(2,8)$, $7(2,5)$
19	$15(3,5)$, $20(2,8)$
$20(t)$	$16(3,7)$, $18(3,8)$, $19(2,8)$
21	$11(3,10)$
22	$14(2,10)$
23	$12(3,8)$
24	$15(2,10)$

A.2 Kathmandu Road Network (cf. Section 4.6)

(i, j)	u_{ij} (per second)	u_{ji} (per second)	τ_{ij}^0 (minutes)
(0, 1)	2	2	6
(0, 12)	2	2	10
(0, 18)	2	2	3
(0, 27)	2	2	4
(1, 2)	2	2	3
(1, 27)	2	2	5
(2, 3)	3	3	5
(2, 38)	3	3	12
(3, 4)	3	3	5
(3, 27)	2	2	6
(4, 5)	3	3	1
(4, 32)	2	2	1
(5, 6)	3	3	1
(5, 42)	2	2	7
(6, 7)	3	3	5
(6, 23)	2	2	2
(7, 8)	3	3	8
(7, 17)	2	2	10
(7, 99)	2	2	5
(8, 9)	2	2	16
(8, 99)	3	3	7
(9, 10)	2	2	3
(9, 17)	2	2	3
(10, 11)	2	2	5
(11, 12)	2	2	17
(11, 37)	2	2	7
(12, 13)	2	2	4
(13, 14)	2	2	6
(13, 33)	2	2	9
(14, 15)	2	2	1
(14, 33)	2	2	3
(15, 16)	2	2	1
(15, 35)	2	2	1
(16, 17)	2	2	1
(16, 36)	2	2	3
(16, 37)	4	0	1

(18, 19)	2	2	2
(18, 28)	2	2	2
(19, 20)	2	2	2
(19, 29)	2	2	2
(20, 21)	2	2	2
(20, 30)	2	2	2
(20, 33)	2	2	1
(21, 26)	0	4	1
(21, 34)	2	2	1
(22, 23)	4	0	1
(22, 32)	2	2	1
(23, 24)	4	0	1
(24, 25)	4	0	2
(25, 26)	4	0	1
(26, 35)	2	2	2
(27, 28)	2	2	3
(28, 29)	4	0	2
(29, 30)	4	0	1
(30, 31)	2	2	2
(31, 32)	2	2	2
(33, 34)	2	2	2
(34, 35)	2	2	1
(35, 36)	2	2	2
(38, 39)	3	3	7
(38, 42)	2	2	8
(39, 40)	3	3	1
(39, 41)	2	2	1
(40, 41)	2	2	2
(40, 99)	3	3	8
(41, 42)	2	2	2

Source: 0, Sink: 99

APPENDIX B

SCIENTIFIC PUBLICATIONS & PRESENTATIONS

B.1 Publications

1. Nath, H. N., Pyakurel, U., Dhamala, T. N., & Dempe, S. (2021). Dynamic network flow location models and algorithms for quickest evacuation planning. *Journal of Industrial and Management Optimization, 17*(5), 2925–2941. doi: 10.3934/jimo.2020102.

2. Nath, H. N., Dempe, S., & Dhamala, T. N. (2021). A bicriteria approach for saving path maximizing dynamic contraflow. *Asia-Pacific Journal of Operational Research*. doi: https://doi.org/10.1142/S0217595921500275.

3. Nath, H. N., Pyakurel, U., & Dhamala, T. N. (2021). Network reconfiguration with orientation-dependent transit times. International Journal of Mathematics and Mathematical Sciences, 2021. doi: 10.1155/2021/6613622.

4. Nath, H. N., & Dhamala, T. N. (2018). Network flow approach for locating optimal sink in evacuation planning. *International Journal of Operations Research, 15*(4), 175–185.

5. Dhamala, T. N., Adhikari, I. M., Nath, H. N., & Pyakurel, U. (2018). Meaningfulness of OR models and solution strategies for emergency planning. In *Living under the threat of earthquakes* (pp. 175–194). Springer.

6. Nath, H. N., Pyakurel, U., Dempe, S., & Dhamala, T. N. (2019a). A bilevel programming approach to save a path maximizing the dynamic flow with lane reversals for evacuation planning. Preprint, TU Bergakademie, Freiberg.

7. Nath, H. N., Pyakurel, U., Dempe, S., & Dhamala, T. N. (2019b). A path saving strategy with arc reversals for evacuation planning. *International Journal of Innovative Knowledge Concepts, 7*(1), 160–165.

8. Nath, H. N., & Dhamala, T. N. (2017). Identification of optimal pick-up

locations with their demands in evacuation planning of transit-dependent population. In *Proceeding of national conference on mathematics and its applications (NCMA–21017)* (pp. 48–53).

9. Nath, H. N., Pyakurel, U., & Dhamala, T. N. (2018). Facility location on arcs for quickest evacuation planning. In *The 11th triennial conference of association of Asia Pacific operational research societies, APORS–2018, abstract and program book* (pp. 115–117).

B.2 Presentations

1. *Optimization Models and Algorithms for Evacuation Planning: The Bus Evacuation Problem*. International Conference on Applications of Mathematics to Nonlinear Sciences (AMNS- 2016), May 26–29, 2016. (Poster)
2. *Meaningfulness of OR Models and Solution Strategies for Emergency Planning*. Mathematics and Statistics Research Colloquium, organized by Mindanao State University, Iligan Institute of Technology, Philippines, 17 August, 2016.
3. *Identification of Optimal Pick-Up Locations with Their Demands in Evacuation Planning of Transit-Dependent Population*. National Conference on Mathematics and Its Applications (NCMA-2017), Chitwan, Nepal, January 11–13, 2017.
4. *Bi-level optimization in Emergency Evacuation Planning*. National Conference on History and Recent Trends of Mathematics (NCHRTM-2017), Kathmandu, Nepal, June 2–4, 2017.
5. *Network Flow Approach for Computing Optimal Sink Location in Evacuation Planning*. 2nd International Conference on Advances in Computational Mathematics, Tribhuvan University, Kathmandu, Nepal, December 23–24, 2018.
6. *A Path Saving Strategy with Arc Reversals for Evacuation Planning*. International Conference on Recent Advances in Informatics, Communication, Management, Health & Applied Sciences (RAICMHAS-2019), Brainware University, Kokata, India, February 2–4, 2019.
7. *A Bilevel Programming Approach to Save a Path Maximizing the Dynamic Flow with Lane Reversals for Evacuation Planning*. 4th International Conference on Dynamics of Disasters (DOD 2019), Kalamata, Greece, July 1–5, 2019.
8. *The Quickest FlowLoc Problem*. International Conference on Computational Sciences - Modeling, Computing and Soft Computing (CSMCS 2020), Department of Mathematics, National Institute of Technology Calicut, Kerala, India.

JOURNAL OF INDUSTRIAL AND
MANAGEMENT OPTIMIZATION

doi:10.3934/jimo.2020102

DYNAMIC NETWORK FLOW LOCATION MODELS AND ALGORITHMS FOR QUICKEST EVACUATION PLANNING

Hari Nandan Nath

Tribhuvan University, Bhaktapur Multiple Campus
Bhaktapur, Nepal

Urmila Pyakurel* and Tanka Nath Dhamala

Central Department of Mathematics, Tribhuvan University
P.O. Box 13143, Kathmandu, Nepal

Stephan Dempe

TU Bergakademie, Fakultät für Mathematik und Informatik
09596 Freiberg, Germany

(Communicated by Alexander Kononov)

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 with a given objective is very important in the optimization of flow in emergency cases. Because of the decrease in 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 propose two polynomial time heuristics for its solution. Because of the growing concerns of arc reversals in evacuation planning, we introduce the quickest ContraFlowLoc problem and present exact algorithms to solve the single-facility case and heuristics to solve the multi-facility case, with 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

2020 *Mathematics Subject Classification.* Primary: 90B10, 90C27, 68Q25; Secondary: 90B06, 90B20.

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

* Corresponding author: Urmila Pyakurel.

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.

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 an 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 a rescue team departs from the rescue center, rides a vehicle towards the rescue transfer center, and walks to each rescuee group to provide an 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 a 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 a combined trip distribution and assignment problem. Ng et al. [19] also use the same approach in which the lower level is a deterministic user equilibrium model as described by Sheffi [35]. In an integrated bus evacuation problem, Goerigk et al. [10] choose the shelters to minimize the maximum travel time over all the buses while in a comprehensive evacuation planning, Goerigk et al. [9] formulate a multiple commodity, multi-criteria problem to minimize the total evacuation time, the risk exposure of evacuees, and the 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 the 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 [32], Hamacher et al. [11], Heller and Hamacher [12] combine the location decisions with the flow decisions in a network flow problem observing that the placement of a facility on an arc of a network may result in a reduction of the maximum flow value. Given a set of facilities and a 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 value is minimum.

The main aim of optimizing the traffic flow in an emergency evacuation process is to maximize the flow and/or to minimize the evacuation time using the road network. The minimum time to transfer a given number of evacuees from disaster areas to safe places, is termed as the quickest time. Placement of facilities in the road segments hampers the traffic flow, resulting in the increase of the quickest time of evacuation. In this paper, our objective is to identify the optimal allocation of facilities to the road-segments so that the evacuation time of a given number of evacuees, to reach the safe places, is minimized. So, our attempt is to contribute to above-mentioned FlowLoc Models by considering the quickest time as a deciding factor of facility location decisions. We also aim at combining such decisions with the reversal of the direction of the traffic flow in some road segments, if necessary.

In emergency 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 segments is increased. Contraflow configuration not only increases flow value but also reduces the traffic jam and makes the traffic smooth [21]. But to identify appropriate directions 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 evacuation time can be reduced by 40% by reversing at most 30% arcs can be found in Kim et al. [14]. In cases, when each node has an associated danger factor, Vogiatzis et al. [39] present a heuristic algorithm to solve the problem of sending vehicles from the nodes with danger factors to the safe nodes, reversing at most a given number of arcs, with an objective to minimize the number of vehicles that have to spend time on the most endangered nodes. They use the smart clustering of similar nodes to create subgraphs so as to solve a large-scale problem efficiently.

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

Pyakurel et al. [26]. The maximum dynamic and the earliest arrival contraflow problems are generalized in Pyakurel et al. [27].

Moreover, the first temporally repeated flow algorithm to solve the quickest contraflow problem, within a complexity of solving a min-cost flow problem, has been presented by Pyakurel et al. [29]. Considering a case of the Kathmandu road network, their comparison of the quickest time, before and after the contraflow configuration, shows that a significant decrease in the quickest time can be attributed to the contraflow configuration and the decrease in the quickest time increases with the number of evacuee-vehicles. They also present an approximate algorithm to solve the quickest contraflow problem with a load-dependent transit time on each arc.

The analytical techniques discussed above use arc-based formulation of network flow problems. Recently a path-based formulation of the similar problem with the abstract flow on abstract networks is also gaining attention. Pyakurel et al. [26] introduce the contraflow technique in abstract networks, present algorithms to solve the maximum static and the maximum dynamic contraflow problems with continuous time setting and realize that if the minimum dynamic cut capacities on a two-terminal network are symmetric, then the flow value can be increased up to double with the 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 the arcs during evacuation process, Pyakurel et al. [28, 30] investigate the partial contraflow problem and present algorithms to solve some related problems.

Motivated by the work of Hamacher et al. [11], our main focus in this paper is to introduce the 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 obstructs 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 are considered in Section 2. Section 3 investigates the quickest FlowLoc problem and presents strongly polynomial algorithms to solve the single facility cases and polynomial heuristics to solve the multiple facility cases. In Section 4, we combine the contraflow decsions with the location decisions on arcs and Section 5 concludes the paper suggesting the 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 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^{in} and the set of arcs going out of it by A_i^{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 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 rate on $e \in A$. For each $i \in V$, we denote the excess of x at $i \in V$ by

$$\text{exc}_x(i) = \sum_{e \in A_i^{\text{in}}} x(e) - \sum_{e \in A_i^{\text{out}}} x(e) \tag{1}$$

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

$$\text{exc}_x(i) = 0, \ \forall i \in V \setminus \{s, d\}. \tag{2}$$

The static flow x is called feasible if

$$0 \leq x(e) \leq b(e), \ \forall e \in A. \tag{3}$$

The value of x is defined as:

$$\text{val}(x) = \text{exc}_x(d). \tag{4}$$

If $\text{exc}_x(i) = 0$ 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 Γ be the collection of all the s-d paths and $\mathcal{C}$ be the collection of cycles of the network. Let $f(\gamma)$ and $f(C)$ be the flows in $\gamma \in \Gamma$ and $C \in \mathcal{C}$, then the arc flow is

$$x(e) = \sum_{\gamma \in \Gamma} \delta_e(\gamma) f(\gamma) + \sum_{C \in \mathcal{C}} \delta_e(C) f(C) \tag{5}$$

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 (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 (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 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 $b_x : A(x) \to \mathbb{R}$ 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.

2.2. **Dynamic flows.**

A dynamic flow Φ with time horizon T consists of Lebesgue-integrable functions $\Phi_e : [0,T) \to \mathbb{R}_{\geq 0}$ 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 θ. The flow entering the tail i of the arc $e = (i,j)$ at time θ 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 Φ at time θ as:

$$\text{exc}_\Phi(i,\theta) = \sum_{e \in A_i^{\text{in}}} \int_0^{\theta-\tau(e)} \Phi_e(\sigma)d\sigma - \sum_{e \in A_i^{\text{out}}} \int_0^{\theta} \Phi_e(\sigma)d\sigma \tag{6}$$

which is the net amount of flow that enters node i up to time θ.

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

$$\text{exc}_\Phi(i,\theta) \geq 0 \ \forall \theta \in [0,T), \ \forall i \in V \setminus \{s\} \tag{7}$$

$$\text{exc}_\Phi(i,T) = 0, \ \forall i \in V \setminus \{s,d\} \tag{8}$$

and

$$0 \leq \Phi_e(\theta) \leq b(e), \ \forall e \in A, \theta \in [0,T). \tag{9}$$

The value of the dynamic flow Φ at time θ is

$$\text{val}_\theta(\Phi) = \text{exc}_\Phi(d,\theta)$$

and the total value of the dynamic flow Φ is:

$$\text{val}(\Phi) = \text{val}_T(\Phi) = \text{exc}_\Phi(d,T)$$

For more details, we refer to Skutella [37].

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 Γ with flow $f(\gamma)$ for each $\gamma \in \Gamma$. Flow is sent along γ 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 γ. In this way, the dynamic flow is obtained as described in the following equation

$$\Phi_e(\theta) = \sum_{\gamma \in \Gamma_e(\theta)} f(\gamma), \ \forall e = (i,j) \in A, \theta \in [0,T), \tag{10}$$

where if $\gamma_{s,i}$ denotes the portion of γ from s to i, and $\gamma_{j,d}$ denotes that from j to d, then $\Gamma_e(\theta) = \{\gamma \in \Gamma | e \in P \text{ and } \tau(\gamma_{s,i}) \leq \theta \text{ and } \tau(\gamma_{j,d}) < T - \theta\}$.

Given a time horizon T, the maximum dynamic problem seeks to find a dynamic flow Φ 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 (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. [33]).

A problem closely related with the maximum dynamic flow problem is the quickest flow problem which seeks to find a 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:*

$$\min \quad \frac{F + \sum_{e\in A} \tau(e)x(e)}{v} \tag{11}$$

$$\text{s.t.} \sum_{e\in A_i^{\text{out}}} x(e) - \sum_{e\in A_i^{\text{in}}} x(e) = \begin{cases} v & if\ i = s \\ -v & if\ i = d \\ 0 & otherwise \end{cases} \tag{12}$$

$$0 \leq x(e) \leq b(e), \ \forall e \in A. \tag{13}$$

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 [34] 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 the 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 $r : P \to \mathbb{N}$ and the number of facilities that can be placed on the possible locations $\nu : L \to \mathbb{N}$. The maximum static FlowLoc problem asks for an allocation $\pi : P \to L$, such that the difference of the maximum static s-d flow value in the network $N^\pi = (V, A, \tau, b^\pi, s, d)$ and that in N is mimized where the capacity function b^π 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 difference of the dynamic s-d flow values is to be minimized, 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 in the increase in the time to transfer a given amount of flow from the source 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 $r : P \to \mathbb{N}$, the number of facilities that can be placed on the possible locations $\nu : L \to \mathbb{N}$, the quickest FlowLoc problem seeks for an allocation $\pi : P \to L$ of the facilities to the arcs, such that the difference of the quickest time to transport F from s to d on the network $N^\pi = (V, A, b^\pi, \tau, s, d)$ and that in N is minimized, where $b^\pi(e) = b(e) - \max\{r(p) : p \in P \text{ and } \pi(p) = e\}$.

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

(i) $|P| \leq \sum_{e\in L} \nu(e)$ 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.

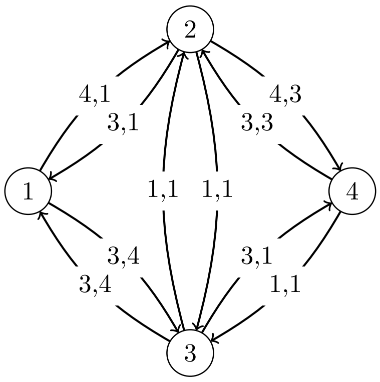


FIGURE 1. Evacuation network N with arc labels (capacity, travel time)

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 1. Maximum dynamic FlowLoc decisions (cf. Figure 1)

T	maximum dynamic flow value when facility is placed on		Location decision
	(2,3)	(2,4)	
4	0	1	(2,4)
5	4	5	(2,4)
6	11	11	(2,4) or (2,3)
7	18	17	(2,3)

TABLE 2. Quickest FlowLoc decisions (cf. Figure 1)

F	quickest time when facility is placed on		Location decision
	(2,3)	(2,4)	
1	4.25	4	(2,4)
5	5.14	5	(2,4)
11	6	6	(2,4) or (2,3)
21	7.43	7.67	(2,3)

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 the cost scaling algorithm by Lin and Jaillet [18]. As expected, the maximum dynamic FlowLoc and quickest FlowLoc decisions are related with each other in some sense.

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

Cost scaling algorithm. 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 ρ 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 ϵ-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 2ϵ-optimal flow is modified to an ϵ-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, ϵ 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 Lin and Jaillet [18].

Cancel-and-tighten algorithm. 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 i, compute the maximum length $h(i)$ from nodes with no entering admissible arc. Replace $\rho(i)$ by $\rho(i) + \frac{\epsilon}{n}h(i)$ and reduce ϵ to $(1 - \frac{1}{n})\epsilon$.

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

The above-mentioned steps are performed until ϵ becomes smaller than $1/4n$, and finally the Saturate step is performed as in the cost scaling algorithm. The complexity of this algorithm is $O(nm^2 \log^2 n)$. For details, we refer to Saho and Shigeno, [34].

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 a preflow push algorithm to perform maximum flow calculations, the time complexity of such an algorithm, as realized in [11], is $O(|L|n^3)$.

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

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

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. [31]), the maximum dynamic flow problem with time horizon T can be solved in $O(mn + m \log m)$ time, using a greedy approach, by sending the flow iteratively through the s-d path with the minimum time and removing the saturated arc, considering only the paths with time not exceeding

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 $e \in L$ do
 4     $b(e) = b(e) - r$
 5     $x'$ = min-cost circulation in $N^c$
 6     $new\_max\_flow = -$cost of $x'$
 7     if $new\_max\_flow > curr\_max\_flow$ then
 8         $curr\_max\_flow = new\_max\_flow$
 9         $loc = e$
10         $x =$ restriction of $x'$ to $N$
11     end
12     $b(e) = b(e) + r$
13 end
14 return $loc, x$
```

T. Thus, in a series parallel graph, the single facility maximum dynamic FlowLoc problem can be solved in $O(|L|(mn + m \log m))$ time. Moreover, in such a graph, a maximum flow has also the earliest arrival property which requires the flow to be maximized at each period of time. Thus, maximum 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.

Now, we construct two algorithms for the single facility quickest FlowLoc problem. Algorithm 2 iterates over all possible locations $e \in L$, determines the quickest time if location e 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 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.

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

Proof. Algorithm 2 performs the quickest flow computation L times and Algorithm 3, in worst case, performs it $|L| + 1$ times. Using the cancel-and-tighten algorithm by Saho and Shigeno [34], 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. □

Algorithm 2: Single facility quickest FlowLoc I

```
Input  : Directed network $N = (V, A, b, \tau, s, d)$, the set of possible locations
         $L$, size $r$ of the single facility, supply $F$ at $s$
Output: Location $loc$ of the facility, the corresponding static flow $x$, the
         corresponding quickest time $T$
1  $T = \infty$
2  for $e \in L$ do
3      $b(e) = b(e) - r$
4      $new\_quickest\_time$ = the quickest time in the modified network
5      if $new\_quickest\_time < T$ then
6          $T = new\_quickest\_time$
7          $loc = e$
8          $x$ = static flow corresponding to the quickest flow
9      end
10     $b(e) = b(e) + r$
11 end
12 return $loc, x, T$
```

Algorithm 3: Single facility quickest FlowLoc II

```
Input  : Directed network $N = (V, A, b, \tau, s, d)$, the set of possible locations
         $L$, size $r$ of the single facility, supply $F$ at $s$
Output: Location $loc$ of the facility, the corresponding static flow $x$, the
         corresponding quickest time $T$
1  $x$ = static flow corresponding to the quickest flow in the network $N$
2  $T$ = the corresponding quickest time
3  $e^* = \arg\max\{b(e) - x(e) : e \in L\}$
4  if $b(e^*) - x(e^*) \geq r$ then
5      $loc = e^*$
6      $b(e^*) = b(e^*) - r$
7  else
8      Algorithm 2
9  end
10 return location $loc, x, T$
```

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 $T = \infty$. 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 at 4. In this way, in 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 α-approximation algorithm for the multi-facility maximum static FlowLoc problem with a finite constant α 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:

$$\max \quad v \tag{14}$$

$$\sum_{e\in A_i^{\text{out}}} x_e - \sum_{e\in A_i^{\text{in}}} x_e = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{if } i \notin \{s, d\} \end{cases} \tag{15}$$

$$0 \le x_e \le b_e\ \forall e \in A \tag{16}$$

The objective function (14) replaced by $\frac{F+\sum_{e\in A} x_e\tau_e}{v}$ 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 (15) and (16). But for a fixed $F > 0$, minimizers of F/v will maximize v. □

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

Proof. Suppose that there is a polynomial time α-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, a polynomial time exact algorithm or an approximate algorithm is not possible. However, in large-scale evacuation situations, a fast sub-optimal solution is preferred to the slow optimal solution (Vogiatzis et al. [40]). So, we design polynomial time heuristics 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.

$$\min \quad \frac{F + \sum_{e\in A} \tau_e x_e}{v} \tag{17}$$

$$\sum_{e\in A_i^{\text{out}}} x_e - \sum_{e\in A_i^{\text{in}}} x_e = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{if } i \in V \setminus \{s,d\} \end{cases} \tag{18}$$

$$x_e + y_{ep} r_p \le b_e, \ \forall e \in L, p \in P \tag{19}$$

$$0 \le x_e \le b_e, \ \forall e \in A \tag{20}$$

$$\sum_{e\in L} y_{ep} = 1, \ \forall p \in P \tag{21}$$

$$\sum_{p\in P} y_{ep} \le \nu_e, \ \forall e \in L \tag{22}$$

$$y_{ep} \in \{0,1\}, \ \forall e \in L, p \in P \tag{23}$$

The variables and constants used in the model are described as follows.

Variables

x_e = static flow rate 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 (18) and (20) are conditions for a static flow. Constraints (19) reduce the capacity of e by r_p if the facility p is placed on e. Constraints (21) state that each facility has to be placed in exactly one arc, and constraints (22) 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. We can make it linear by putting $1/v = \theta$, and $x_e\theta = \xi_e$. As a result the problem becomes

$$\min \quad F\theta + \sum_{e\in A} \tau_e \xi_e \tag{24}$$

$$\sum_{e\in A_i^{\text{out}}} \xi_e - \sum_{e\in A_i^{\text{in}}} \xi_e = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = d \\ 0 & \text{if } i \in V \setminus \{s,d\} \end{cases} \tag{25}$$

$$\xi_e + \theta y_{ep} r_p \le b_e\theta, \ \forall e \in L, p \in P \tag{26}$$

$$0 \le \xi_e \le b_e\theta, \ \forall e \in A \tag{27}$$

$$\sum_{e\in L} y_{ep} = 1, \ \forall p \in P \tag{28}$$

$$\sum_{p\in P} y_{ep} \le \nu_e, \ \forall e \in L \tag{29}$$

$$y_{ep} \in \{0,1\}, \ \forall e \in L, p \in P \tag{30}$$

However, the set of constraints (26) 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 [38], replacing (26) with the following constraints $\forall e \in L, p \in P$

$$\begin{aligned} \xi_e + \zeta_{ep} r_p &\leq b_e \theta && (31)\\ \zeta_{ep} &\leq M y_{ep} && (32)\\ \zeta_{ep} &\leq \theta && (33)\\ \zeta_{ep} &\geq \theta - (1 - y_{ep})M && (34)\\ \zeta_{ep} &\geq 0 && (35) \end{aligned}$$

where M is an upper bound on the values of θ 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 present two polynomial time heuristics, in Algorithm 4 and 5, to solve the problem. In Algorithm 4, 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) 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 I

```
   Input  : Directed network $N = (V, A, b, \tau, s, d)$, supply $F$ at the source $s$,
            the set of possible locations $L$ with number of facilities $\nu : L \to \mathbb{N}$,
            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 the facilities in $P$, according to the size, as $p_1, p_2, \cdots, p_q$ such that
     $r(p_1) \geq r(p_2) \geq \cdots \geq r(p_q)$
 2 $x$ = static flow corresponding to the quickest flow
 3 $T$ = the corresponding quickest time
 4 $k = 1$
 5 while $k \leq q$ do
 6     $e^* = \arg\max\{b(e) - x(e) : e \in L\}$
 7     for $l = 1$ to $l = \nu(e^*)$ do
 8         if $k + l - 1 \leq q$ then
 9             $\pi(p_{k+l-1}) = e^*$
10         end
11     end
12     $L = L \setminus \{e^*\}$
13     $b(e^*) = b(e^*) - r(p_k)$
14     if $b(e^*) - x(e^*) + r(p_k) < r(p_k)$ then
15         $x$ = static flow corresponding to the quickest flow with modified $b$
16         $T$ = the corresponding quickest time
17     end
18     $k = k + \nu(e^*)$
19 end
20 return $\pi, x, T$
```

Example 3. To illustrate Algorithm 4, we consider the network given in Figure 1. Let $F = 11$, $L = \{(2,1),(2,4),(1,3),(3,4)\}$ 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
$b(e) - x(e) : e \in L$	1	3	1				0			

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

First iteration:
$e^* = \arg\max\{b(e) - x(e) : e \in L\} = (2,1)$
$l = 1$
$k + l - 1 = 1 + 1 - 1 = 1$
$\pi(p_1) = (2,1)$
$L = \{(2,4),(1,3),(3,4)\}$
$b(e^*) = 3 - 3 = 0$
$b(e^*) - x(e^*) + r(p_k) = 0 - 0 + 3 \not< r(p_1)$
$k = 1 + 1 = 2 < q$

Second iteration:
$e^* = \arg\max\{b(e) - x(e) : e \in L\} = (2,4)$ (We may take $e^* = (1,3)$ also.)
$l = 1, 2$
$k + l - 1 = 2 + 1 - 1, 2 + 2 - 1 = 2, 3$
$\pi(p_2) = (2,4)$
$\pi(p_3) = (2,4)$
$L = \{(1,3),(3,4)\}$
$b(e^*) = 4 - 2 = 2$
$b(e^*) - x(e^*) + r(p_k) = 2 - 3 + 2 = 1 < r(p_k) = 2$
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
$b(e) - x(e) : e \in L$	1						0			

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

Third iteration:
$e^* = \arg\max\{b(e) - x(e) : e \in L\} = (1,3)$
$l = 1$
$k + l - 1 = 4 + 1 - 1 = 4$
$\pi(p_4) = (1,3)$
$L = \{(3,4)\}$
$b(e^*) = 3 - 1 = 2$

$b(e^*) - x(e^*) + r(p_k) = 2 - 2 + 1 = 1 \not< r(p_k) = r(p_4) = 1$
$k = 4 + 1 = 5 > q$
Since $k > q$, 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 the 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.

In the above example, if we solve the single-facility quickest FlowLoc problem each time when we require to place a facility in a new arc, the objective function value matches with that of the MILP solution. We present such a procedure to solve a multi-facility quickest FlowLoc problem in Algorithm 5. Although the running time of the algorithm is higher, because of more quickest flow calculations, the objective function values are closer to the optimal values in most of the cases in the computational experiment done in Section 3.3.

Algorithm 5: Multi facility quickest FlowLoc heuristic II

```
Input  : Directed network $N = (V, A, b, \tau, s, d)$, supply $F$ at the source $s$,
         the set of possible locations $L$ with number of facilities $\nu : L \to \mathbb{N}$,
         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 the facilities in $P$, according to the size, as $p_1, p_2, \cdots, p_q$ such that
     $r(p_1) \geq r(p_2) \geq \cdots \geq r(p_q)$
2  $k = 1$
3  while $k \leq q$ do
4      $e^*$ = optimal location obtained by solving the single facility quickest
         FlowLoc problem for $p_k$
5      for $l = 1$ to $l = \nu(e^*)$ do
6          if $k + l - 1 \leq q$ then
7              $\pi(p_{k+l-1}) = e^*$
8          end
9      end
10     $L = L \setminus \{e^*\}$
11     $b(e^*) = b(e^*) - r(p_k)$
12     $k = k + \nu(e^*)$
13 end
14 $x$ = static flow corresponding to the quickest flow with modified $b$
15 $T$ = the corresponding quickest time
16 return $\pi, x, T$
```

3.3. **Computational experiment.** To test the performance of the proposed heuristics, we take a transportation network of Kathmandu city, consisting of major road segments within the Ring Road (Figure 2a). We take Dasharath Stadium and neighboring area as the source. The stadium is the largest in Nepal and the neighboring area consists of some of the major shopping malls. We consider a scenario of evacuating people in case of an emergency, e.g. the threat of a possible bomb attack, from the source to the outside of the Ring Road.

The directed graph representing the network consists of 69 nodes and 221 arcs. The travel time on each arc is considered to be the time of travel on the corresponding road segment at the speed of 40 km/h. The length of each road segment is based on the OpenStreetMap data. Depending on the width of the road segment, the capacity of the corresponding arc is taken from 1 to 4 units of flow (an average sized vehicle) per second.

Given a flow value F, we calculate the quickest flow and identify the arcs with nonzero flow (see Figure 2b for $F = 20000$) and consider a random set of feasible locations L to contain at least 25% such arcs, because if L consists entirely of zero-flow arcs, the solution is trivial. The set of feasible facilities P is also taken randomly with admissible sizes. For each $e \in L$, the number of facilities it can host is chosen randomly between 1 to a maximum of 2 facilities per kilometer. We focus, mainly, on the solutions with objective function values differing from the quickest time without allocating facilities.

For $F = 20000$, outcomes of some typical instances are presented in Table 4. The corresponding quickest time without facility allocation is 2570 seconds. The running time (R. time), and the objective function values (T^*) are expressed in seconds. The MILP solutions with running time more than 15 minutes are not recorded.

For $F = 5000, 20000$ and 50000, with at least 30 instances each (for which the MILP solutions are recorded), the maximum and average percentage deviations from the MILP objective function value are shown in Table 3.

TABLE 3. Percentage Deviation from the MILP objective function values

	Algorithm 4	Algorithm 5
Maximum deviation	21.55%	5.31 %
Average deviation	3.48 %	0.18%

Out of 178 instances (including the above-mentioned instances), only 16 instances of Algorithm 4 have an objective value better than that of Algorithm 5. However, the running time of Algorithm 5 is higher than that of Algorithm 4. Among the tested instances, the maximum running time of Algorithm 4 is 0.59 seconds with an average of 0.17 seconds, the corresponding values for Algorithm 5 are 2.31 seconds and 1.02 seconds.

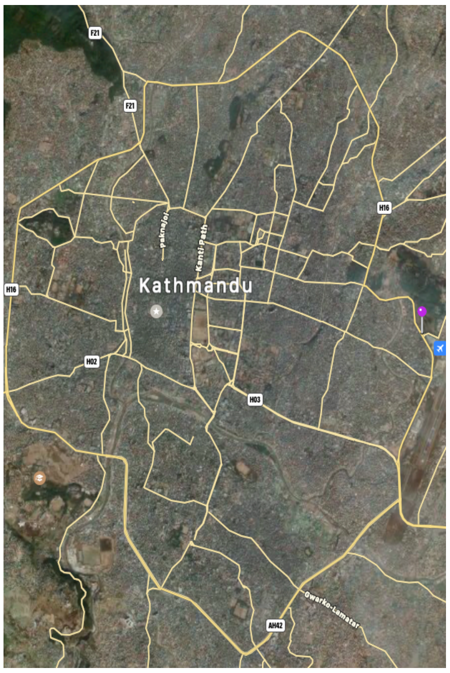
The implementation of the algorithms and the mixed integer programming are done using the programming language Python 3.7 on a computer with Mac operating system having 1.8 GHz dual-core Intel Core i5 processor, and 8 GB RAM. The solver used to solve the mixed integer program is CBC (Coin-OR branch and cut).

4. Quickest FlowLoc problem with contraflow.

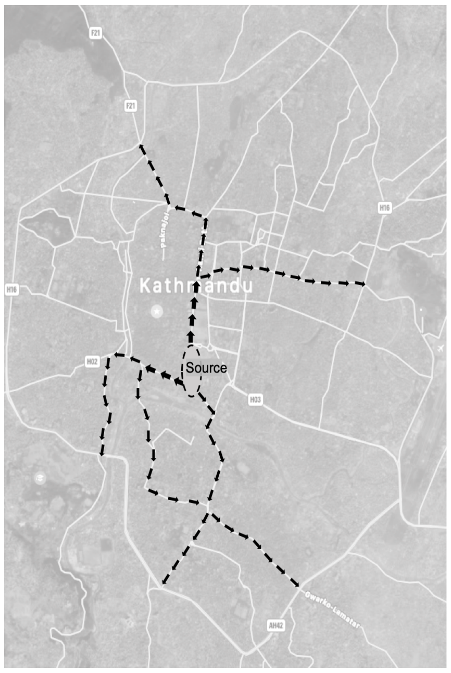
Assuming that the direction of arcs in a directed network can be reversed (i.e. the direction of the traffic flow

TABLE 4. Computational results for some instances with $F = 20000$

$\lvert L\rvert$	$\lvert P\rvert$	Algorithm 4		Algorithm 5		MILP	
		R. time	T^*	R. time	T^*	R. time	T^*
5	5	0.14	3209.3	0.53	2717.6	0.87	2713.5
5	7	0.16	2831.9	0.49	2707.6	1.39	2707.1
6	7	0.13	3222.1	0.65	2881.2	1.93	2878.1
8	10	0.15	2580.0	0.58	2575.0	2.92	2575.0
8	8	0.14	3189.3	0.47	3189.3	2.01	3189.3
9	10	0.15	2725.9	0.66	2593.3	3.46	2593.3
10	10	0.17	2587.8	0.73	2585.6	8.01	2585.6
11	10	0.11	2847.5	0.81	2573.9	5.69	2573.9
11	11	0.17	2586.7	0.83	2580.6	15.16	2580.6
11	15	0.20	2721.8	0.72	2710.6	52.52	2707.6
12	11	0.15	2708.2	0.73	2585.0	6.7	2585.0
12	15	0.17	2877.5	0.69	2881.2	114.77	2877.5
13	15	0.12	2588.3	0.51	2576.1	2.82	2576.1
14	15	0.24	2578.3	0.72	2579.4	645.67	2576.1
14	21	0.11	2861.9	0.69	2712.4	134.76	2595.0
15	17	0.16	3005.3	0.98	2843.1	224.36	2843.1
15	20	0.16	3036.7	1.10	2733.5	47.37	2733.5
16	17	0.16	2736.5	1.06	2722.4	221.88	2722.4
17	19	0.19	2851.9	1.29	2717.1	379.54	2716.5
18	19	0.20	2581.7	0.97	2583.3	214.78	2581.1
18	19	0.16	2589.4	1.18	2585.6	822.64	2585.6
19	20	0.24	2595.0	1.93	2607.2	-	-
19	24	0.59	3032.7	1.76	2868.8	-	-
20	25	0.18	2707.6	1.25	2707.6	-	-
21	25	0.15	2577.8	1.21	2577.8	-	-
23	25	0.14	2692.9	1.47	2575.0	-	-
23	26	0.14	2705.3	1.05	2705.3	-	-
24	26	0.20	3026.0	1.87	2863.1	-	-
26	29	0.14	2704.7	1.42	2704.7	-	-
28	40	0.20	2585.0	1.14	2585.0	-	-
28	39	0.17	2586.7	1.20	2586.7	-	-
29	34	0.16	2851.2	1.42	2851.2	-	-
30	40	0.24	2588.3	1.55	2588.3	-	-
31	38	0.17	2851.2	1.75	2851.2	-	-
31	37	0.22	2600.0	1.57	2600.0	-	-
32	40	0.17	2706.5	1.54	2706.5	-	-
37	43	0.22	2701.8	1.77	2701.8	-	-
38	41	0.21	2597.2	1.44	2598.9	-	-
39	50	0.25	2710.0	1.95	2710.0	-	-
40	50	0.09	2573.3	1.76	2571.7	-	-
41	48	0.20	2710.6	2.19	2710.6	-	-



(a) Kathmandu network



(b) Quickest flow arc occupancy (F=20000)

FIGURE 2

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

$$\bar{A} = \{(i, j) : (i, j) \in A \text{ or } (j, i) \in A\}$$

For each $(i, j) \in \bar{A}$,

$$\begin{aligned} \bar{b}(i,j) &= b(i,j) + b(j,i) \\ \bar{\tau}(i,j) &= \begin{cases} \tau(i,j) & \text{if } (i,j) \in A \\ \tau(j,i) & \text{otherwise} \end{cases} \end{aligned}$$

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 [23, 31]. 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. [29]. The algorithms to solve the contraflow problems not only give optimal flow decisions but also declare 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.

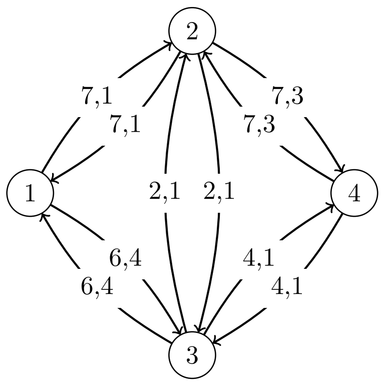


FIGURE 3. Auxiliary network $\bar{N}$ of network N in Figure 1 with arc labels (capacity, travel time)

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, has 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 and $\nu : L \to \mathbb{N}$, 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.

To solve the single-facility maximum static(dynamic) ContraFlowLoc 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

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

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

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 6 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 6: 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 $(i,j) \in L$ do
3      $b(i,j) = b(i,j) - r$
4      $new\_quickest\_time$ = quickest contraflow time in the modified network
         if $new\_quickest\_time < T$ then
5          $T = new\_quickest\_time$
6          $loc = (i,j)$
7          $x$ = the corresponding static contraflow
8      end
9      $b(i,j) = b(i,j) + r$
10 end
11 $R = \{(j,i) \in A : x(i,j) > b(i,j)$ if $(i,j) \in A$ or $x(i,j) > 0$ if $(i,j) \notin A\}$
12 return $loc, x, T, R$
```

We can improve the running time of Algorithm 6 by adapting Algorithm 3 to the contraflow case, in the cases when there is enough capacity in a feasible arc to hold the given facility. After contraflow calculation, if arc $(i,j) \in L$ and its opposite arc $(j,i) \in A$ together have capacity enough to host the facility, then (i,j) is chosen to locate the facility and we can get rid of the remaining $|L| - 1$ quickest contraflow calculations of Algorithm 6. The procedure is elucidated in Algorithm 7.

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. [29]). 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 $(i,j) \in L$ reduces the capacity of (i,j) and (j,i) both in the auxiliary network, Algorithm 6

Algorithm 7: 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 in $N$
2  $T$ = the corresponding quickest time
3  $(i^*, j^*) = \arg\max\{b(i,j) + b(j,i) - x(i,j) - x(j,i) : (i,j) \in L\}$
4  if $b(i^*, j^*) + b(j^*, i^*) - x(i^*, j^*) - x(j^*, i^*) \geq r$ then
5      $loc = (i^*, j^*)$
6      $b(i^*, j^*) = b(i^*, j^*) - r$
7  else
8      Algorithm 6
9  end
10 $R = \{(j,i) \in A : x(i,j) > b(i,j)$ if $(i,j) \in A$ or $x(i,j) > 0$ if $(i,j) \notin A\}$
11 return $loc, x, T, R$
```

and Algorithm 7 find the location *loc* with the minimum quickest time. Moreover, the removal of cycle flows in a contraflow computation leads either $x(i,j)$ or $x(j,i)$ to vanish so that the set R of arcs to be reversed is well defined. This discussion leads to the following lemma:

Lemma 4.3. *Algorithm 6 or Algorithm 7 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 6 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 6 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, beause solving the problem in the auxiliary network is not easier than solving the corresponding problem in the original network. 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 8.

Further, solving the single-facility quickest ContraFlowLoc problem each time when we require to place a facility in a new arc, we can easily adapt Algorithm 5 to the cotraflow case also.

Example 5. To illustrate Algorithm 8, we reconsider the problem illustrated in Example 3 with the possibility of arc reversals. The static contra flow corresponding

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

to the quickest contra flow is tabulated in the following table. For simplicity, we write $b(i,j) + b(j,i) - x(i,j) - x(j,i)$ as $\delta(i,j)$.

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
$\delta(i,j) : (i,j) \in L$	1	3	1				0			

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

First iteration:

$(i^*, j^*) = \max\{\delta(i,j) : (i,j) \in L\} = (1,3)$
$l = 1$
$k + l - 1 = 1 + 1 - 1 = 1$
$\pi(p_1) = (1,3)$
$L = \{(2,1), (2,4), (3,4)\}$
$b(i^*, j^*) = 3 - 3 = 0$

$\delta(i^*, j^*) + r(p_k) = 0 + 3 - 2 - 0 + 3 = 4 \not< r(p_1) = 3$
$k = 1 + 1 = 2 < q$
Second iteration: $(i^*, j^*) = \max\{\delta(i,j)|(i,j) \in L\} = (2,4)$
$l = 1, 2$
$k + l - 1 = 2 + 1 - 1, 2 + 2 - 1 = 2, 3$
$\pi(p_2) = (2,4)$
$\pi(p_3) = (2,4)$
$L = \{(2,1), (3,4)\}$
$b(i^*, j^*) = 4 - 2 = 2$
$d(i^*, j^*) + r(p_k) = 2 + 3 - 5 - 0 + 2 = 2 \not< r(p_2) = 2$
$k = 2 + 2 = 4 = q$

Third iteration:
$(i^*, j^*) = \max\{\delta(i,j) : (i,j) \in L\} = (2,1)$
$l = 1$
$k + l - 1 = 4 + 1 - 1 = 4$
$\pi(p_4) = (2,1)$
$L = \{(3,4)\}$
$b(i^*, j^*) = 3 - 1 = 2$
$d(i^*, j^*) + r(p_k) = 2 + 4 - 7 - 0 + 1 = 0 < r(p_4) = 1$
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

$k = 4 + 1 = 5 > q$

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 network flow models, some 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 a single facility and polynomial time heuristic algorithms to place multiple facilities on multiple arcs are designed. The related algorithms in literature, so far, either find the quickest flow in the network without facility assignment or assign facilities to the arcs without quickest flow considerations. Our proposed algorithms combine quickest flow with facility assignment and find the optimal (or near optimal) assignment so that the increase in the quickest time is the least after facility allocation to the arcs. 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.

The performance of heuristic algorithms are tested taking the Kathmandu road network as an evacuation network. In randomly generated instances of the set of feasible locations and set of facilities, the faster heuristic shows an average deviation of approximately 3.5%, and the slower 0.2% from the actual optimal solution. The corresponding algorithms allowing arc reversals are also presented. As in the absence of facility allocations, a significant improvement in the quickest time because of necessary arc reversals is observed in the FlowLoc case also. 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.

However, in the problems considered, the location 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 natural extensions of the problem.

Acknowledgments. The first author acknowledges UGC Nepal for the PhD fellowship support, and TU Bergakademie Freiberg for the research stay support (April-September 2019). The second author acknowledges the support of Alexander von Humboldt (AvH) Foundation for her post doctoral research stay (November 2017-October 2019) at TU Bergakademie, Freiberg, Germany and her return fellowship (November 2019-October 2020) at Central Department of Mathematics, TU, Nepal. The authors thank AvH for the support of the AvH Research Group Linkage Program between TU Bergakademie Freiberg, Germany and Central Department of Mathematics, TU Kathmandu, Nepal. The authors would also like to thank the anonymous reviewers for their constructive suggestions.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, New Jersey, 1993.

[2] S. An, N. Cui, X. Li and Y. Ouyang, Location planning for transit-based evacuation under the risk of service disruptions, *Transportation Research Part B: Methodological*, **54** (2013), 1–16.

[3] R. D. Armstrong and Z. Jin, A new strongly polynomial dual network simplex algorithm, *Mathematical Programming*, **78** (1997), 131–148.

[4] T. N. Dhamala and U. Pyakurel, Earliest arrival contraflow problem on series-parallel graphs, *International Journal of Operations Research*, **10** (2013), 1–13.

[5] T. N. Dhamala, U. Pyakurel and S. Dempe, A critical survey on the network optimization algorithms for evacuation planning problems, *International Journal of Operations Research*, **15** (2018), 101–133.

[6] R. C. Dhungana and T. N. Dhamala, Maximum FlowLoc Problems with Network Reconfiguration, *International Journal of Operations Research*, **16** (2019), 13–26.

[7] R. C. Dhungana, U. Pyakurel and T. N. Dhamala, Abstract contraflow models and solution procedures for evacuation planning, *Journal of Mathematics Research*, **10** (2018), 89–100.

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

[9] M. Goerigk, K. Deghdak and P. Heßler, A comprehensive evacuation planning model and genetic soution algorithm, *Transportation Research, Part E*, **71** (2014), 82–97.

[10] M. Goerigk, B. Grün and P. Heßler, Combining bus evacuation with location decisions: A branch-and-price approach, *Transportation Research Procedia*, **2** (2014), 783–791.

[11] H. W. Hamacher, S. Heller and B. Rupp , Flow location (FlowLoc) problems: Dynamic network flows and location models for evacuation planning, *Annals of Operations Research*, **207** (2013), 161–180.

[12] S. Heller and H. W. Hamacher, The multi-terminal q-FlowLoc problem: A heuristic, in *Lecture Notes in Computer Science, Proceedings of the International Network Optimization Conference*, Springer, **6701** (2011), 523–528.

[13] H. Jia, F. Ordóñez and M. Dessouky, A modeling framework for facility location of medical services for large-scale emergencies, *IIE Transactions*, **39** (2007), 41–55.

[14] S. Kim, S. Shekhar and M. Min, Contraflow transportation network reconfiguration for evacuation route planning, *IEEE Transactions on Knowledge and Data Engineering*, **20** (2008), 1–15.

[15] S. Kongsomsaksakul, C. Yang and A. Chen, Shelter location-allocation model for flood evacuation planning, *Journal of the Eastern Asia Society for Transportation Studies*, **6** (2005), 4237–4252.

[16] E. Köhler, K. Langkau and M. Skutella, Time expanded graphs for flow-dependent transit times, in *European Symposium on Algorithms* (eds. R. Möhring and R. Raman), Springer, **2461** (2002), 599–611.

[17] A. Kulshrestha, Y. Lou and Y. Yin, Pick-up locations and bus allocation for transit-based evacuation planning with demand uncertainty, *Journal of Advanced Transportation*, **48** (2014), 721–733.

[18] M. Lin and P. Jaillet, On the quickest flow problem in dynamic networks–a parametric min-cost flow approach, in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, (2015), 1343–1356.

[19] M. Ng, J. Park and S. T. Waller, A hybrid bilevel model for the optimal shelter assignment in emergency evacuations, *Computer-Aided Civil and Infrastructure Engineering*, **25** (2010), 547–556.

[20] J. B. Orlin, Max flows in $O(nm)$ time, or better, in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, (2013), 765–774.

[21] U. Pyakurel, *Evacuation Planning Problem with Contraflow Approach*, PhD thesis, IOST, Tribhuvan University, Nepal, 2016.

[22] U. Pyakurel and T. N. Dhamala, Models and algorithms on contraflow evacuation planning network problems, *International Journal of Operations Research*, **12** (2015), 36–46.

[23] U. Pyakurel and T. N. Dhamala, Continuous time dynamic contraflow models and algorithms, *Advances in Operations Research - Hindawi*, **2016** (2016), Art. ID 7902460, 7 pp.

[24] U. Pyakurel and T. N. Dhamala, Continuous dynamic contraflow approach for evacuation planning, *Annals of Operations Research*, **253** (2017), 573–598.

[25] U. Pyakurel and T. N. Dhamala, Evacuation planning by earliest arrival contraflow, *Journal of Industrial and Management Optimization*, **13** (2017), 487–501.

[26] U. Pyakurel, T. N. Dhamala and S. Dempe, Efficient continuous contraflow algorithms for evacuation planning problems, *Annals of Operations Research*, **254** (2017), 335–364.

[27] U. Pyakurel, H. W. Hamacher and T. N. Dhamala, Generalized maximum dynamic contraflow on lossy network, *International Journal of Operations Research Nepal*, **3** (2014), 27–44.

[28] U. Pyakurel, H. N. Nath, S. Dempe and T. N. Dhamala, Efficient dynamic flow algorithms for evacuation planning problems with partial lane reversal, *Mathematics*, **7** (2019), 993.

[29] U. Pyakurel, H. N. Nath and T. N. Dhamala, Efficient contraflow algorithms for quickest evacuation planning, *Science China Mathematics*, **61** (2018), 2079–2100.

[30] U. Pyakurel, H. N. Nath and T. N. Dhamala, Partial contraflow with path reversals for evacuation planning, *Annals of Operations Research*, **283** (2019), 591–612.

[31] S. Rebennack, A. Arulselvan, L. Elefteriadou and P. M. Pardalos, Complexity analysis for maximum flow problems with arc reversals, *Journal of Combinatorial Optimization*, **19** (2010), 200–216.

[32] B. Rupp, *FlowLoc: Discrete Facility Locations in Flow Networks*, Diploma thesis, University of Kaiserslautern, Germany, 2010.

[33] S. Ruzika, H. Sperber and M. Steiner, Earliest arrival flows on series-parallel graphs, *Networks*, **57** (2011), 169–173.

[34] M. Saho and M. Shigeno, Cancel-and-tighten algorithm for quickest flow problems, *Network*, **69** (2017), 179–188.

[35] Y. Sheffi, *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*, Prentice-Hall, Englewood Cliffs, 1985.

[36] H. D. Sherali, T. B. Carter and A. G. Hobeika, A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions, *Transportation Research Part B: Methodological*, **25** (1991), 439–452.

[37] M. Skutella, An introduction to network flows over time, in *Research Trends in Combinatorial Optimization*, (2009), 451–482.

[38] E. Torres, Linearization of mixed-integer products, *Mathematical Programming*, **49** (1990), 427–428.

[39] C. Vogiatzis, J. L. Walteros and P. M. Pardalos, Evacuation through clustering techniques, in *Models, Algorithms, and Technologies for Network Analysis*, Springer New York, **32** (2013), 185–198.

[40] C. Vogiatzis, R. Yoshida, I. Aviles-Spadoni, S. Imamoto and P. M. Pardalos, Livestock evacuation planning for natural and man-made emergencies, *International Journal of Mass Emergencies and Disasters*, **31** (2013), 25–37.

Received January 2019; revised January 2020.

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

Asia-Pacific Journal of Operational Research
(2021) 2150027 (24 pages)
© World Scientific Publishing Co. & Operational Research Society of Singapore
DOI: 10.1142/S0217595921500275

A Bicriteria Approach for Saving a Path Maximizing Dynamic Contraflow

Hari Nandan Nath
Central Department of Mathematics, Tribhuvan University
P.O. Box 13143, Kathmandu, Nepal
hari672@gmail.com

Stephan Dempe
Fakultät für Mathematik und Informatik
TU Bergakademie Freiberg, 09596 Freiberg, Germany
dempe@math.tu-freiberg.de

Tanka Nath Dhamala*
Central Department of Mathematics
Tribhuvan University, P.O. Box 13143
Kathmandu, Nepal
amb.dhamala@daadindia.org

Received 9 September 2020
Revised 16 April 2021
Accepted 8 May 2021
Published

The maximum dynamic contraflow problem in transportation networks seeks to maximize the flow from a source to a sink within a given time horizon with a possibility of arc reversals. This may result into blockage of paths of desired length from some node of the network towards the source. In some cases such as the evacuation planning, we may require a path towards the source to move some facilities, for example, emergency vehicles. In this work, we model the problem of saving such a path as a bicriteria optimization problem which minimizes the length of the path and maximizes the dynamic flow with arc reversals. We use the ϵ-constraint approach to solve the problem and propose a procedure that gives the set of all Pareto optimal solutions in a single-source-single-sink network with integer inputs. We also present computational performance of the algorithm on a road network of Kathmandu city, and on randomly generated networks. The results are of both theoretical and practical importance.

Keywords: Network flow; contraflow; multicriteria optimization; bicriteria optimization; dynamic flow.

*Corresponding author.

1. Introduction

"Network flows" is a topic studied in various fields including applied mathematics, engineering, computer science, management, and operations research, with wide applications (Ahuja *et al.*, 1993). One of the important applications is in evacuation planning to optimize the traffic flow in a complex traffic network to save life in various disasters. We refer to the survey by Dhamala *et al.* (2018) and citations therein for various optimization problems making use of network flow theory to develop algorithms related to evacuation planning. In such problems, road segments are taken as arcs, their intersections (including hazardous areas, and safe areas) are taken as nodes of a network (graph). The nodes corresponding to the hazardous areas are called sources, and those corresponding to the safe areas are called sinks. Most of the problems deal with minimizing the travel time, minimizing the cost or maximizing the value of the flow from the sources to the sinks.

One of the important strategies in evacuation planning problems is flow optimization with arc reversals — known as contraflow approach in literature. The idea is to reverse the usual direction of the traffic flow in necessary road segments to optimize the flow. In a multi-source-multi-sink network, Kim *et al.* (2008) model the problem of minimizing the evacuation time of a given number of evacuees, as an integer programming formulation. Showing that the problem is NP-hard, they propose two heuristics to solve the problem. In a single-source-single-sink network, however, Rebennack *et al.* (2010) show that the problem of maximizing the flow value within a given time horizon, and minimizing the evacuation time of a given number of evacuees at the source can be solved in strongly polynomial times, and present exact algorithms to solve the problems. Pyakurel and Dhamala (2015) solve the earliest arrival contraflow problem on a two-terminal network in pseudo-polynomial time. They solve it in strongly polynomial time if the network is series parallel. The continuous time solution is given in Pyakurel and Dhamala (2016) and the solutions with similar objectives, are solved in Pyakurel *et al.* (2017, 2019).

The contraflow approach, undoubtedly, increases the flow value towards the sink and decreases the evacuation time. However, it may obstruct the paths towards the source. To facilitate evacuation, it may be imperative to use the arcs to place the facilities (Hamacher *et al.*, 2013; Nath *et al.*, 2021). A path saving strategy, in such cases, to save a path not exceeding a given length (travel time) to maximize the flow is modeled in Nath *et al.* (2019b). A related problem with bilevel modeling is presented in Nath *et al.* (2019a).

In the evacuation process of affected people in emergencies, evacuees move away from emergency location and the emergency facilities have to be moved in the opposite direction, from their location towards the emergency. Our motivation of this work comes from the work of Hamacher *et al.* (2013) in which they mention the death and injuries of a significant number of people which can be attributed to the lack of proper planning of these opposite movement scenarios (e.g., more than 500 people were injured and 21 people died in a LoveParade (Germany) disaster in the summer of 2010). Given a set of road segments, and a set of facilities, Hamacher *et al.* (2013) introduce static and dynamic FlowLoc models to allocate facilities to hamper the maximum flow the least. A similar problem to find the quickest time is introduced in Nath *et al.* (2021). However, these models do not identify a path to move the facilities towards the disastrous locations. Reserving a path towards the disastrous locations may hamper the flow of the evacuees towards the safe areas. So, one has to choose the path so as to minimize the loss of the flow. However, such a path may be too long to reach the source within the desired time. Hence, there arises a problem of maximizing the flow of evacuees and minimizing the length of the path to move the facilities, hand in hand. To address the issue, in the present work, we model the problem of saving a path towards the source (disastrous location) as a bicriteria optimization model, in which one objective minimizes the path length from a given node towards the source, and the other objective maximizes the flow value (allowing arc reversals) towards the sink without using the arcs in the path chosen. We present a solution procedure based on ϵ-constrained approach (expressing the objective of minimizing the path length as a constraint bounded by ϵ) and test the computational performance of the procedure in different randomly generated networks and in a portion of Kathmandu (Nepal) city road network.

The paper is organized as follows. Basic ideas required for the work are presented in Sec. 2. The modeling of the problem is presented in Sec. 3. A solution strategy is described in Sec. 4, followed by computational experiments in Sec. 5. Section 6 concludes the paper.

2. Basic Ideas

A network $N = (V, A, \Phi)$ is a directed graph where V is a finite set of nodes with $|V| = n$, A is a set of arcs or edges with $|A| = m$, and Φ is a set of attributes associated with V, A. Without loss of generality, we assume that there are at most two arcs between each pair of nodes $i, j \in V$. The arc e directed from i to j is denoted by (i, j) and that from j to i by (j, i). The node i is called the tail of e and j, the head of e. For each $i \in V$, we define the set of nodes immediately succeeded by i as

$$V_i^+ = \{j \in V : (i, j) \in A\}$$

and the set of nodes immediately preceded by i as

$$V_i^- = \{j \in V : (j, i) \in A\}.$$

With a view that some commodity flows from some nodes of a network to some other nodes, a travel time function

$$\tau : A \to \mathbb{R}_{\geq 0}$$

and a capacity function

$$u : A \to \mathbb{R}_{\geq 0}$$

are defined on A. The travel time $\tau(i,j) = \tau_{ij}$ can be considered as the time the flow takes to reach from i to j, and the capacity $u(i,j) = u_{ij}$, as the maximum amount of flow that can arrive at j from i per unit time. We will denote a network N with the set of nodes V, the set of arcs A, the capacity function u, and the travel time function τ by $N = (V, A, u, \tau)$.

2.1. *Static flow*

Given two distinct nodes $s, t \in V$, a static $s - t$ flow is defined as a function $x : A \to \mathbb{R}_{\geq 0}$ that satisfies the flow conservation

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = 0, \quad \forall\, i \in V \backslash \{s,t\} \tag{1}$$

and the *capacity constraints*

$$0 \leq x_{ij} \leq u_{ij}, \quad \forall\, (i,j) \in A \tag{2}$$

where $x(i,j) = x_{ij}$. The value $v(x)$ of the flow x is defined as

$$v(x) = \sum_{j \in V_s^+} x_{sj} - \sum_{j \in V_s^-} x_{js} = \sum_{j \in V_t^-} x_{jt} - \sum_{j \in V_t^+} x_{tj} \tag{3}$$

which is the total amount of the flow that leaves s to reach t in a single wave without consideration of travel time on arcs. The node s is termed as the *source* node, t as the sink node, and the nodes other than s, t as the intermediate nodes. If there is no ambiguity, a static $s - t$ flow is termed simply as a static flow.

Chain flow

Let P be an $s - t$ chain (a directed path) in N, a chain flow x^P of value $\delta > 0$ on P is a static flow such that

$$x_{ij}^P = \begin{cases} \delta & \text{if } (i,j) \in P, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Cycle flow

If C is a directed cycle in N, a cycle flow x^C with value $\delta > 0$ on C is a circulation such that

$$x_{ij}^C = \begin{cases} \delta & \text{if } (i,j) \in C, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Flow decomposition

We can decompose any static flow into chain and cycle flows. Although the number of chains and cycles in a network is not polynomial in the number of nodes and arcs, there are at most $m+n$ chain and cycle flows in such a decomposition (Ahuja *et al.*, 1993).

Theorem 2.1 (Ahuja *et al.*, 1993). *Let $\mathcal{P}$ be the set of $s-t$ chains and $\mathcal{C}$ be the set of all cycles in N. A static flow x can be decomposed into at most $m+n$ (positive) flows on $s-t$ chains and directed cycles such that*

$$x_{ij} = \sum_{P \in \mathcal{P}\,:\,(i,j)\in P} x_{ij}^P + \sum_{C\in\mathcal{C}:(i,j)\in C} x_{ij}^C$$

with at most m non-zero cycle flows.

2.2. *Dynamic flow*

Given a time horizon $T \geq 0$, a dynamic flow $f = (f_{ij})_{(i,j)\in A}$ where $f_{ij} : [0,T) \to \mathbb{R}_{\geq 0}$ is a Lebesgue measurable function such that

$$f_{ij}(\theta) = 0 \quad \text{for } \theta \geq T - \tau_{ij}. \tag{6}$$

The quantity $f_{ij}(\theta)$ represents the rate of flow that enters i at time θ and reaches j at time $\theta+\tau_{ij}$. So the outflow rate at j at time $\theta+\tau_{ij}$ is $f_{ij}(\theta)$. Hence, the outflow rate at any time θ' at j is $f_{ij}(\theta'-\tau_{ij})$. We define the excess of the node i at time θ as

$$\mathrm{ex}_f(i,\theta) = \sum_{j\in V^-(i)} \int_0^{\theta-\tau_{ji}} f_{ji}(\xi)d\xi - \sum_{j\in V^+(i)} \int_0^{\theta} f_{ij}(\xi)d\xi \tag{7}$$

which is the net amount of flow that enters the node i up to time θ. Given two distinct nodes $s,t \in V$, a dynamic flow f is called a dynamic $s-t$ flow if

$$\mathrm{ex}_f(i,\theta) \geq 0, \quad \forall\, i \in V\backslash\{s\}, \quad \theta \in [0,T), \tag{8}$$

$$\mathrm{ex}_f(i,T) = 0, \quad \forall\, i \in V\backslash\{s,t\}, \tag{9}$$

$$0 \leq f_{ij}(\theta) \leq u_{ij}, \quad \forall\, (i,j) \in A, \quad \theta \in [0,T). \tag{10}$$

The value of the $s-t$ flow over time is

$$v^T(f) = \mathrm{ex}_f(t,T). \tag{11}$$

Constraints (8) and (9) allow to store flow at intermediate nodes for some time as long as it has left the node again before the time horizon is over.

2.2.1. *Temporally repeated flow*

Given a feasible static flow x and a time horizon T, a flow decomposition on x gives a set of paths $\mathcal{P}$ with flow x_P for each $P \in \mathcal{P}$. If a flow is sent along P at a constant rate x_P from the source during the time interval $[0, T - \tau(P))$, we can obtain a flow over time with time horizon T, where $\tau(P) = \sum_{(i,j)\in P} \tau_{ij}$ is the travel time on path P. For a node i in an $s - t$ path P, let P_{si}, P_{it} denote the subpaths of P from s to i and from i to t, where

$$\mathcal{P}_{ij}(\theta) = \{P \in \mathcal{P} : (i,j) \in P \text{ with } \tau(P_{si}) \leq \theta \text{ and } \tau(P_{it}) < T - \theta\}.$$

Then a temporally repeated flow f is obtained as described in the following equation:

$$f_{ij}(\theta) = \sum_{P\in\mathcal{P}_{ij}(\theta)} x_P, \quad \forall\, (i,j) \in A, \quad \theta \in [0,T). \tag{12}$$

The temporally repeated flow described in (12) is a dynamic $s - t$ flow over time, with strict equality in the constraints (8), i.e., it does not allow flow to store at the intermediate nodes (Skutella, 2009).

2.2.2. *Maximum dynamic flow*

For a given network N with source s, sink $t \neq s$, and a time horizon $T \geq 0$, a dynamic flow f^* is called a maximum dynamic flow if $v^T(f^*) \geq v^T(f)$ for any dynamic $s - t$ flow f defined in N. So, the maximum dynamic flow problem can be stated as

$$\max v^T(f)$$

$$\text{subject to (8)–(10).}$$

The construction of maximum dynamic flow from static flows by Ford and Fulkerson (1958) gives a rather simplified version of the problem as a linear program (see also Fleischer and Tardos (1998))

$$\min \; -Tv + \sum_{(i,j)\in A} \tau_{ij} x_{ij}, \tag{13a}$$

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s, \\ -v & \text{if } i = t, \\ 0 & \text{if } i \in V\backslash\{s,t\}, \end{cases} \tag{13b}$$

$$0 \leq x_{ij} \leq u_{ij}, \quad \forall\, (i,j) \in A, \tag{13c}$$

which gives static flow x, of value v, temporal repetition of which yields the maximum dynamic flow f with the value

$$v^T(f) = Tv - \sum_{(i,j)\in A} \tau_{ij} x_{ij}.$$

2.3. *Contraflow models*

Contraflow models seek to identify a set of arcs $R \subseteq A$ so that each $(i,j) \in R$ can be reversed to optimize the flow. One of the important strategies to solve such a problem is to solve the corresponding flow problem in a modified network known as auxiliary network.

Auxiliary network

The auxiliary network of $N = (V, A, u, \tau)$, is defined as the network $N' = (V, A', u', \tau')$, where

$$A' = \{(i,j) : (i,j) \in A \text{ or } (j,i) \in A\},$$

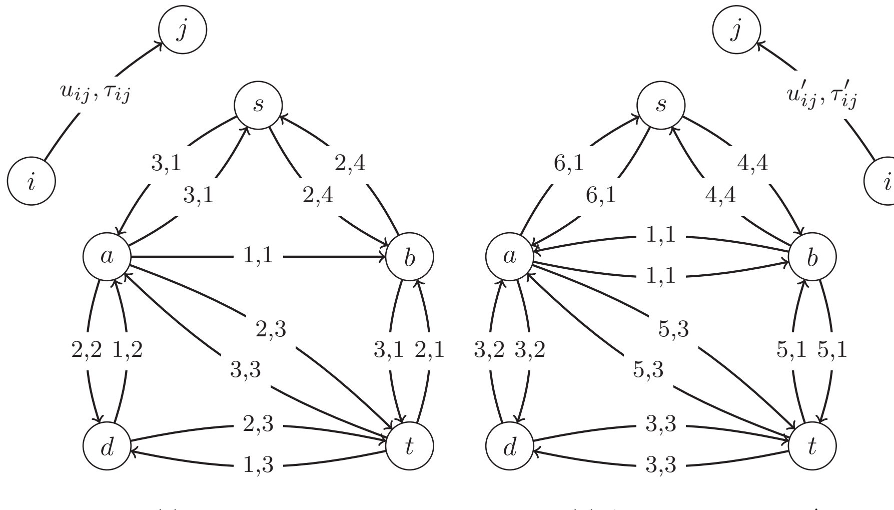
the capacity function $u' : A' \to \mathbb{R}_{\geq 0}$ is given by

$$u'(i,j) = u'_{ij} = \begin{cases} u_{ij} & \text{if } (j,i) \notin A, \\ u_{ji} & \text{if } (i,j) \notin A, \\ u_{ij} + u_{ji} & \text{otherwise,} \end{cases}$$

and the travel time function $\tau' : A \to \mathbb{R}_{\geq 0}$ is given by

$$\tau'(i,j) = \tau_{ij} = \begin{cases} \tau_{ij} & \text{if } (i,j) \in A, \\ \tau_{ji} & \text{otherwise.} \end{cases}$$

This concept is depicted in Fig. 1. Each arc label represents the capacity and the travel time associated with the corresponding arc.



(a) Network N (b) Auxiliary network N'

Fig. 1. Auxiliary network construction.

2.4. *Multicriteria optimization*

Let $\psi_i : \mathcal{X} \to \mathbb{R}, i = 1, \ldots, p$, then a problem of the form

$$\begin{aligned} &\min(\psi_1(x), \ldots, \psi_p(x)) \\ &\text{subject to } x \in \mathcal{X} \end{aligned} \tag{14}$$

is called multicriteria optimization problem, where $\mathcal{X}$ is the feasible set. The minimization (min) has not be considered in the ordinary sense because elements of $\mathbb{R}^p$ are not ordered in the sense the elements in $\mathbb{R}$ are ordered. We define some special orders in $\mathbb{R}^p$ as follows.

Componentwise order in $\mathbb{R}^p$

For $a = (a_1, \ldots, a_p), b = (b_1, \ldots, b_p) \in \mathbb{R}^p$, the following orders in $\mathbb{R}^p$ are defined.

(i) $a \leqq b \Leftrightarrow a_1 \leq b_1, \ldots, a_p \leq b_p$ (weak componentwise order),
(ii) $a \leq b \Leftrightarrow a_1 \leq b_1, \ldots, a_p \leq b_p, a \neq b$ (componentwise order),
(iii) $a < b \Leftrightarrow a_1 < b_1, \ldots, a_p < b_p$ (strict componentwise order).

Pareto optimality, efficiency, weak efficiency

Let $\psi = (\psi_1, \ldots, \psi_p)$. For $a, b \in \mathcal{X}$, if $\psi(a) \leq \psi(b)$, we say that a dominates b, and $\psi(a)$ dominates $\psi(b)$. A feasible solution $a^* \in \mathcal{X}$ is called Pareto optimal or efficient if there is no $a \in \mathcal{X}$ that dominates a^*. In other words, a^* is Pareto optimal if there is no $a \in \mathcal{X}$ satisfying $\psi(a) \leq \psi(a^*)$. A feasible solution $a^w \in \mathcal{X}$ is called weakly Pareto optimal (weakly efficient) if there is no $a \in \mathcal{X}$ satisfying $\psi(a) < \psi(a^w)$.

It is obvious that an efficient solution is also a weakly efficient solution, but the converse is not true. The following result is important to find the solutions of the bi-criteria model developed in this work.

Theorem 2.2 (Ehrgott, 2005). *An optimal solution of the problem*

$$\begin{aligned} &\min_{x \in \mathcal{X}} \psi_j(x) \\ &\text{subject to } \psi_k(x) \leq \epsilon_k, k = 1, \ldots, p \quad k \neq j, \end{aligned} \tag{15}$$

where $\epsilon \in \mathbb{R}^p$, *is a weakly efficient solution of the problem* (14).

3. Developing the Model

3.1. *The maximum dynamic contraflow problem*

Given a time horizon T, the maximum dynamic contraflow problem seeks to identify the maximum dynamic contraflow with a possibility of reversing a set of arcs. Rebennack *et al.* (2010) introduce the problem and present a procedure to solve it. We present the procedure in Algorithm 1.

Algorithm 1. Maximum dynamic contraflow algorithm (Rebennack *et al.*, 2010)

```
Input  : A network $N = (V, A, u, \tau)$, source $s$, sink $t$, time horizon $T$
Output: Maximum dynamic flow with arc reversal on $N$
1 Construct the auxiliary network $N'$ of $N$.
2 Calculate the static flow $x$ corresponding to the temporally repeated
  maximum dynamic flow on $N'$.
3 Decompose $x$ into path (chain) flows and cycle flows. Remove cycle flows and
  update $x$.
4 Arc $(j, i) \in A$ is reversed if and only if $(i, j) \in A$ and $x_{ij} > u_{ij}$ or $(i, j) \notin A$
  and $x_{ij} > 0$.
5 Maximum dynamic flow with arc reversal on $N$ = temporally repeated flow
  generated by $x$.
```

Example 3.1. Consider a network given in Fig. 1(a) with source s, sink t and time horizon $T = 10$. The value of the maximum dynamic flow is 29 as given in the following table.

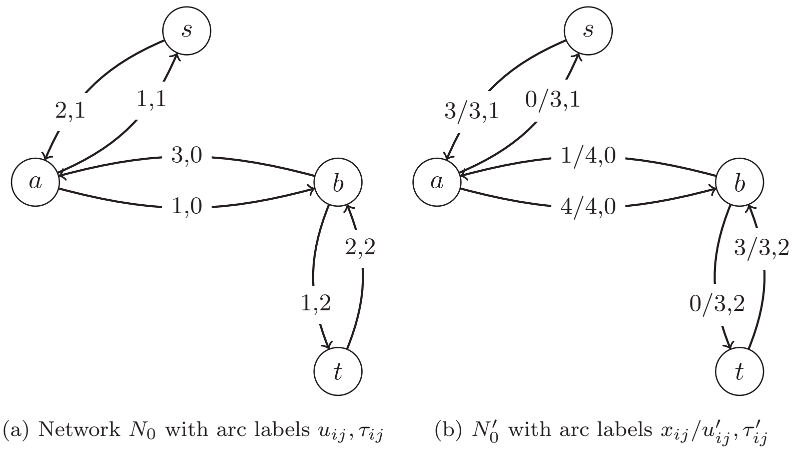
Path (P)	x^P	$T - \tau(P)$	Dynamic flow value
s–a–b–t	1	7	7
s–a–t	2	6	12
s–b–t	2	5	10
Total dynamic flow value			29

Solving the problem in the auxiliary network (Fig. 1(b)), the value of the maximum dynamic flow is 57 as mentioned in the following table. The arcs to be reversed being $(a, s), (b, s), (t, a), (t, b)$.

Path (P)	x'^P	$T - \tau'(P)$	Dynamic flow value
s–a–b–t	1	7	7
s–a–t	5	6	30
s–b–t	4	5	20
Total dynamic flow value			57

Removal of the cycle flows in Step 3 of Algorithm 1 is crucial to maintain the feasibility of the flow in the network N after arc reversals. The following example illustrates this fact.

Example 3.2. Consider a network as depicted in Fig. 2(a). Its auxiliary network is given in Fig. 2(b) with static flow x corresponding to temporally repeated maximum dynamic flow with $T \geq 4$. We can decompose x into a path flow of value 3 along s–a–b–t and a cycle flow of value 1 along a–b–a. Removing the cycle flow, we get a feasible flow in the network constructed from N_0 with arcs $(a, s), (b, a), (b, t)$ reversed. If we do not remove the cycle flow, it is feasible in the auxiliary network but not feasible



(a) Network N_0 with arc labels u_{ij}, τ_{ij} (b) N_0' with arc labels $x_{ij}/u'_{ij}, \tau'_{ij}$

Fig. 2. The network considered in Example 3.2.

in network N_0 even after arc reversals. In N_0', the value v of the static flow is 3 which remains unchanged even after the removing the cycle flow. Taking $T = 5$, the value of the dynamic flow is $v^T(f) = Tv - \sum \tau_{ij} x_{ij} = 5 \times 3 - 3 - 6 = 6$. If $\tau_{ab} > 0$, and x is the same, then removing the cycle flow does not change v but $v^T(f)$ is increased.

The following theorem shows that if there is no cycle of zero length in N (the length of cycle C is $\sum_{(i,j)\in C} \tau_{ij}$), there does not exist a positive flow in any of the cycles of N.

Theorem 3.1. *If every cycle in N is of positive length, an optimal flow in the solution of problem* (13) *does not have a positive flow in a cycle in its flow decomposition.*

Proof. Suppose that $D(x) = -Tv + \sum_{(i,j)\in A} \tau_{ij} x_{ij}$. Let x^* be the optimal solution of problem (13) with $v = v(x^*) = v^*$ and the flow decomposition of x^* have a positive flow in cycles. Assume C to be the set of arcs which form a cycle with flow value $\delta > 0$, and that

$$\sum_{(i,j)\in C} \tau_{ij} > 0.$$

Define $x^1, x^2 : A \to \mathbb{R}$ by

$$x^1_{ij} = \begin{cases} x^*_{ij} & \text{for } (i,j) \in A \backslash C, \\ x^*_{ij} - \delta & \text{for } (i,j) \in C, \end{cases}$$

and x^2 be a flow defined by

$$x^2_{ij} = \begin{cases} 0 & \text{for } (i,j) \in A \backslash C, \\ \delta & \text{for } (i,j) \in C. \end{cases}$$

Then x^1 and x^2 are feasible static flows in N such that $x^1_{ij} + x^2_{ij} = x^*_{ij}$, $\forall\,(i,j) \in A$. Moreover, since a static flow in a cycle does not contribute to the value of the static flow, $v(x^1) = v^*$. Now,

$$\begin{aligned} D(x^*) &= -Tv^* + \sum_{(i,j)\in A} \tau_{ij} x^*_{ij} \\ &= -Tv^* + \sum_{(i,j)\in A} \tau_{ij} x^1_{ij} + \sum_{(i,j)\in A} \tau_{ij} x^2_{ij} \\ &= -Tv^* + \sum_{(i,j)\in A} \tau_{ij} x^1_{ij} + \delta \sum_{(i,j)\in C} \tau_{ij} \\ &> -Tv^* + \sum_{(i,j)\in A} \tau_{ij} x^1_{ij} \left(\because \delta > 0, \sum_{(i,j)\in C} \tau_{ij} > 0 \right) = D(x^1). \end{aligned}$$

Since x^1 is a also a feasible static flow with $v(x^1) = v^*$, this contradicts that x^* is optimal because $x^1 \neq x^*$. □

An immediate corollary of the above theorem is.

Corollary 3.1. *Given $\tau_{ij} > 0$, $\forall\,(i,j) \in A$, if x^* is an optimal solution of problem* (13), *there is no positive cycle flow in its flow decomposition.*

If no positive cycle flow can be guaranteed beforehand, Step 3 of Algorithm 1 can be skipped. Then the problem of finding the temporally repeated maximum dynamic flow in the auxiliary network can be expressed as the linear program (13) with A replaced by A', u replaced by u', and τ replaced by τ'. In this way, because of Theorem 3.1, we get the following result.

Theorem 3.2. *In a network $N = (V, A, u, \tau)$, if*

(a) *for each $(i,j) \in A$, there is $(j,i) \in A$ with $\tau_{ij} = \tau_{ji}$,*
(b) *every cycle in N is of positive length,*

then a solution given by Algorithm 1 *is also a solution of the linear program*

$$\min\ -Tv + \sum_{(i,j)\in A} \tau_{ij} x_{ij}, \tag{16a}$$

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = \begin{cases} v & \text{if } i = s, \\ -v & \text{if } i = t, \\ 0 & \text{if } i \in V\backslash\{s,t\}, \end{cases} \tag{16b}$$

$$0 \le x_{ij} \le u_{ij} + u_{ji}, \forall (i,j) \in A, \tag{16c}$$

with (i,j) reversed if $x_{ji} > u_{ji}$.

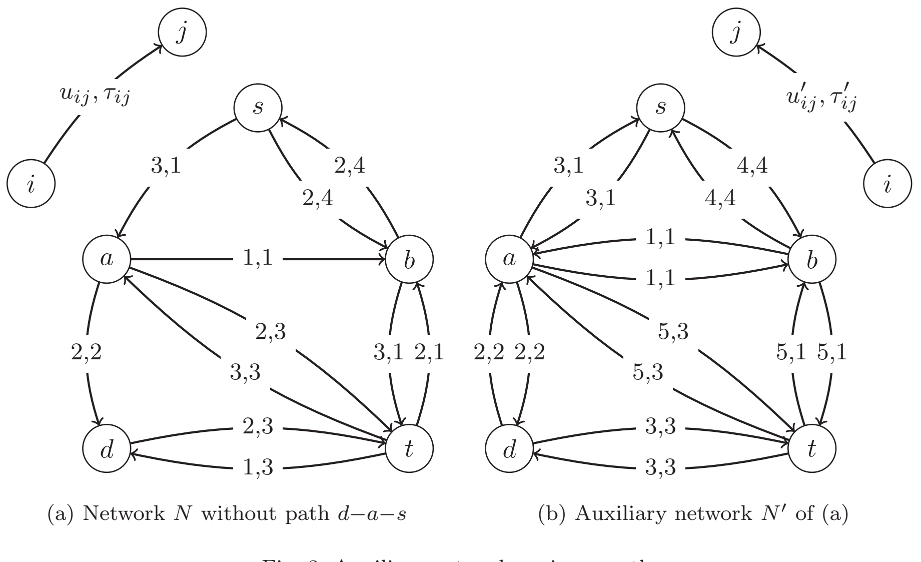
Assumption (a) in the above theorem is non-restrictive, because if $(j, i) \notin A$ for some $(i, j) \in A$, we may add (j, i) to A with $u_{ji} = 0$ and $\tau_{ji} = \tau_{ij}$.

The arc reversal in a network increases the value of the flow whenever the reversal results in the increase of the capacity of a path towards the sink. This is useful, particularly, in case of evacuation planning. However, it may obstruct a flow towards the source. In this work, we consider a problem of reserving a path (so that there is no flow on any of its arcs) from a specific node, called depot, to the source for some kind of facility movements.

Example 3.3. Consider the network in Fig. 1(a), in which s is the source, t is the sink, and d is a depot. If we keep the path d–a–s for a possible facility movement (Fig. 3(a)), taking $T = 10$, the maximum dynamic flow value with arc reversals (solving the maximum dynamic flow problem in the auxiliary network 3(b)) is 39 (1 through path s–a–b–t repeated 7 times, 2 through s–a–t repeated 6 times, and, 4 through s–b–t repeated 5 times).

The dynamic flow values and the lengths for all the possible paths are given in the following table.

Saved path (P)	$\tau(P)$	$v^{10}(f)$
$P_1 : d$–a–s	3	39
$P_2 : d$–t–a–s	7	39
$P_3 : d$–a–b–s	7	44
$P_4 : d$–t–b–s	8	47
$P_5 : d$–a–t–b–s	10	43
$P_6 : d$–t–a–b–s	11	26



(a) Network N without path d−a−s (b) Auxiliary network N' of (a)

Fig. 3. Auxiliary network saving a path.

If we consider only the maximum flow value, the path P_4 is the optimal path with maximum flow value 47. But sometimes, the length of the path also matters, e.g. to move the facility the fastest. The possible candidate for this option is the path P_1. Considering the objectives of minimizing the path length and maximizing flow value, the non-dominated (see Sec. 2.4) paths are P_1, P_3, and P_4.

3.2. *Bicriteria path-saving model to maximize dynamic contraflow* ($BPMDC$)

In this section, we model the problem of saving a path towards the source from a specific node called depot with the following assumptions.

(1) If there is an arc (i, j) between the nodes i and j, there is an arc (j, i) with a positive travel time equal to that of (i, j). If there is no arc (j, i) corresponding to (i, j), we can construct one with zero capacity. The symmetric travel time is assumed so that the contraflow modeling of Rebennack *et al.* (2010) can be applied. Positive travel time is considered to use the model presented in Theorem 3.2.
(2) Each path from the depot to the source (without a zero capacity arc) has enough capacity for the movement of facilities so that each path is a candidate for the path to be saved. See Remark 3.1 for a more general case.
(3) Once a path is chosen to be saved, we maximize the dynamic flow from the source to the sink in the rest of the network allowing arc reversals without using any of the arcs in the saved path.
(4) For making the model simple, we neither allow arc reversals in the paths to be saved, nor we use any unused capacity of the arcs in the saved path for the dynamic flow.

We require the saved path to be as short as possible, and the value of the maximum dynamic contraflow as large as possible. As these objectives are conflicting, in general, there is no single solution that optimizes both the objectives simultaneously. So, we formulate the problem as a bicriteria optimization problem.

Consider a network $N = (V, A, u, \tau)$ in which $(j, i) \in A$ whenever $(i, j) \in A$, and $\tau_{ij} = \tau_{ji}$. We formulate the problem on the basis of the famous shortest path problem formulation and Theorem 3.2. Considering $s, t, d \in V$ as the source, the sink, and the depot, the variables and the parameters are described below.

Variables
x_{ij} : static flow rate on the arc (i, j),

$$y_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is in the saved path,} \\ 0 & \text{otherwise.} \end{cases}$$

Parameters
T : time horizon, a positive integer,

τ_{ij} : travel time on the arc (i,j), a positive integer,
u_{ij} : capacity of the arc (i,j), a non-negative integer.

Assuming that

$$\psi_1 = \sum_{(i,j)\in A} \tau_{ij} y_{ij}$$

and

$$\psi_2 = -T\left[\sum_{j\in V_s^+} x_{sj} - \sum_{j\in V_s^-} x_{js}\right] + \sum_{(i,j)\in A} \tau_{ij} x_{ij},$$

the problem is formulated as:

$$\min(\psi_1, \psi_2) \tag{17a}$$

subject to:

$$\sum_{j\in V_i^+} y_{ij} - \sum_{j\in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s, \\ 0 & \text{if } i \in V\backslash\{s,d\}, \\ 1 & \text{if } i = d, \end{cases} \tag{17b}$$

$$y_{ij} \le u_{ij}, \quad \forall (i,j) \in A, \tag{17c}$$

$$\sum_{j\in V_i^+} x_{ij} - \sum_{j\in V_i^-} x_{ji} = 0, \quad \forall i \in V\backslash\{s,t\}, \tag{17d}$$

$$0 \le x_{ij} \le (1-y_{ij})u_{ij} + (1-y_{ji})u_{ji}, \quad \forall (i,j) \in A, \tag{17e}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A. \tag{17f}$$

The objective (17a), minimizes the path length and the negative of the value of the dynamic flow with path reversal. Constraints (17b) are the constraints for the saved path. Constraints (17c) ensure that the path chosen does not contain the arcs with zero capacity. Constraints (17d) are flow-conservation constraints, and constraints (17e) limit the static flow rate on an arc (i,j) by u_{ij} if (j,i) is in the saved path, by u_{ji} if (i,j) is in the saved path, and by $u_{ij} + u_{ji}$ otherwise. For brevity, we refer to the model as BPMDC.

Remark 3.1. Each arc of the path saved by the above model will have capacity at least one. However, if one needs a path with a capacity at least $b \in \mathbb{Z}_{>0}$, one can define

$$U_{ij} = \begin{cases} 1, & b \leq u_{ij}, \\ 0, & b > u_{ij}, \end{cases}$$

and replace the constraint (17c) by

$$y_{ij} \leq U_{ij}, \quad \forall\, (i,j) \in A. \tag{17g}$$

This will force $y_{ij} = 0$ when $b > u_{ij}$. Consequently, the arcs with capacity less than b will not be chosen for the saved path.

4. Solution Strategy

To solve the problem, we apply the idea given in Theorem 2.2 to convert the problem into an ϵ-constrained mixed binary integer linear program which gives a weakly Pareto optimal (weakly efficient) solution for each $\epsilon \in \mathbb{R}$ and then develop a procedure that gives Pareto optimal solutions whenever the transit time function is positive integer-valued. We formulate the ϵ-constrained problem ($\epsilon \in \mathbb{R}$) as

$$\min -T\left[\sum_{j \in V_s^+} x_{sj} - \sum_{j \in V_s^-} x_{js}\right] + \sum_{(i,j) \in A} \tau_{ij} x_{ij}, \tag{18a}$$

subject to

$$\sum_{j \in V_i^+} y_{ij} - \sum_{j \in V_i^-} y_{ji} = \begin{cases} -1 & \text{if } i = s, \\ 0 & \text{if } i \in V \backslash \{s,d\}, \\ 1 & \text{if } i = d, \end{cases} \tag{18b}$$

$$y_{ij} \leq u_{ij}, \quad \forall\, (i,j) \in A, \tag{18c}$$

$$\sum_{j \in V_i^+} x_{ij} - \sum_{j \in V_i^-} x_{ji} = 0, \quad \forall\, i \in V \backslash \{s,t\}, \tag{18d}$$

$$0 \leq x_{ij} \leq (1 - y_{ij})u_{ij} + (1 - y_{ji})u_{ji}, \quad \forall\, (i,j) \in A, \tag{18e}$$

$$y_{ij} \in \{0,1\}, \tag{18f}$$

$$\sum_{(i,j) \in A} \tau_{ij} y_{ij} \leq \epsilon. \tag{18g}$$

In Algorithm 2, we propose a procedure which not only lists out the paths corresponding to the non-dominated values of (ψ_1, ψ_2) but also gives the corresponding static flow that can be temporally repeated to get the corresponding maximum dynamic flow.

Algorithm 2. Non-dominated solutions of BPMDC

```
Input  : $N = (V, A, u, \tau)$ with $u_{ij} \in \mathbb{Z}_{\geq 0}, \tau_{ij} \in \mathbb{Z}_{>0}$, a source $s$, a sink $t$, a
         depot $d$
Output: Set of non-dominated saved paths with the maximum dynamic
         contraflow
1  $\epsilon_{\min}$ = length of a shortest $d$–$s$ path, $\epsilon_{\max} = \sum_{(i,j)\in A} \tau_{ij}$
2  LIST = {}
3  $\psi_2^0 = -\infty$
4  $\epsilon_1 = \epsilon_{\max} + 1$
5  $k = 1$
6  while $\epsilon_k > \epsilon_{\min}$ do
7      $X_k$ = solution of problem (18) for $\epsilon = \epsilon_k - 1$
8      add $X_k$ to LIST
9      $\psi_1^k = \psi_1(X_k)$, $\psi_2^k = \psi_2(X_k)$
10     if $\psi_2^k = \psi_2^{k-1}$ then
11         remove $X_{k-1}$ from LIST
12     end
13     $\epsilon_{k+1} = \psi_1^k$
14     $k = k + 1$
15 end
16 For each $X = \{(x_{ij})_{(i,j)\in A}, (y_{ij})_{(i,j)\in A}\} \in LIST$, construct the path using
   $(i, j)$ with $y_{ij} = 1$ and the dynamic flow by temporal repetition of $x$.
```

Theorem 4.1. *Given $\tau_{ij} \in \mathbb{Z}_{>0}$, let $\mathcal{X}$ be the feasible set of BPMDC* (17)*. Let $\mathcal{Y}_D$ be the set of all non-dominated points in $\psi(\mathcal{X})$, and $S = \{(\psi_1(X), \psi_2(X)) : X \in LIST\}$, then $S = \mathcal{Y}_D$.*

Proof. Because of Theorem 2.2, for each X_k given in Line 7 of Algorithm 2 $(\psi_1(X_k), \psi_2(X_k))$ is weakly non-dominated in $\psi(\mathcal{X})$. Because ϵ decreases by at least 1 in each iteration, the sequence $\{\psi_1(X_k)\}$ is a strictly decreasing sequence. So because of Lines 10 and 11, $(\psi_1(X_k), \psi_2(X_k)) \in \mathcal{Y}_D$.

Conversely, let $X^* \in \mathcal{X}$ such that $(\psi_1(X^*), \psi_2(X^*)) \in \mathcal{Y}_D$ and $(\psi_1(X^*), \psi_2(X^*)) \notin S$. Clearly, the value of ψ_1 (a d–s path length) is bounded above by $\sum_{(i,j)\in A} \tau_{ij}$ and below by the length of the shortest path. Thus, if $LIST = \{X_1, \ldots, X_p\}$, there exists l such that $\psi_1(X_l) > \psi_1(X^*) > \psi_1(X_{l+1})$ and $\psi_2(X_l) < \psi_2(X^*) < \psi_2(X_{l+1})$, where $1 \leq l < p$. This implies that there exists a d–s path of length $\psi_1(X^*)$ in N saving which gives a maximum dynamic contraflow value $-\psi_2(X^*)$. So, putting $\epsilon = \psi_1(X^*)$ in problem (18) gives the minimum value of ψ_2 as $\psi_2(X^*)$. This is a contradiction, because for such an ϵ the minimum value of ψ_2 is $\psi_2(X_{l+1})$ according to Algorithm 2. □

Although integrality restrictions are not put for the variable x in problem (17), if u, τ are integral, one can guarantee the existence of solutions in which x is integral.

Theorem 4.2. *Given* $u_{ij} \in \mathbb{Z}_{\geq 0}, \tau_{ij} \in \mathbb{Z}_{\geq 0}$, *there exist solutions to problem* (17) *with integral* x.

Proof. Fixing y_{ij} to satisfy constraints (17b) and (17f), the problem is equivalent to a maximum dynamic flow problem in the auxiliary network resulting from the removal of a d–s path. Since a maximum dynamic flow problem with integer inputs always has an integral solution, the result follows. $\square$

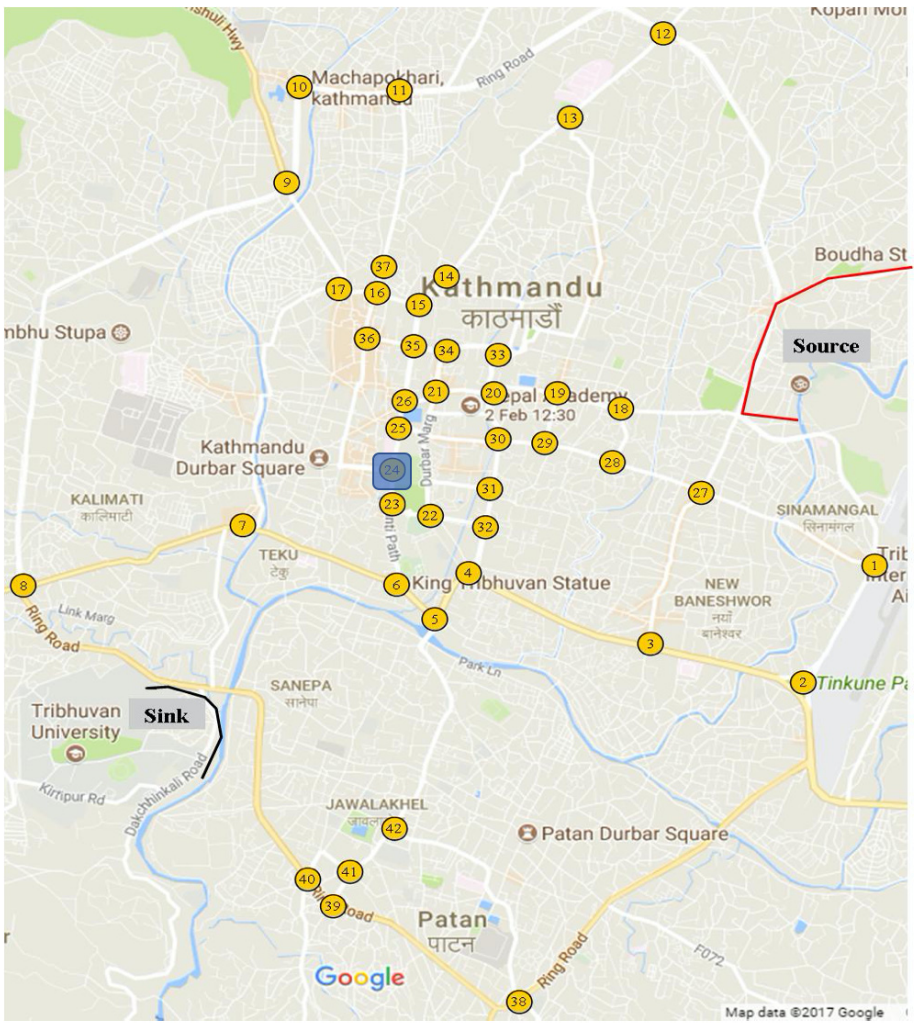


Fig. 4. Kathmandu road network for the computational experiment (Google, n.d.).

The constraints (17b) and (17f) are satisfied by a $d - s$ path which may include cycles (subtours). However, the paths in the solutions generated by Algorithm 2 do not contain such cycles. Let us consider a path with a cycle. Removing the cycle decreases the value of ψ_1. However, it does not decrease the corresponding maximum dynamic contraflow value because removal of the cycle frees capacities which can be used by the flow. This leads to:

Theorem 4.3. *The saved paths in the solution of problem* (17) *are simple.*

5. Computational Experiments

We test the computational performance of the proposed algorithm, first, considering a road network of Kathmandu (Nepal), and then considering randomly generated networks of different sizes as N.

5.1. *Kathmandu road network*

We consider a portion of Kathmandu road network inside the Ring Road considering only the major road-segments (no. of nodes $n = 44$, no. of arcs $m = 132$). We take Pashupati Nath region as a source node s, where a large gathering of people takes place in various religious occasions, and Tribhuvan University region as a sink node t, where there is a sufficient open space (see Fig. 4). For an auto-based evacuation

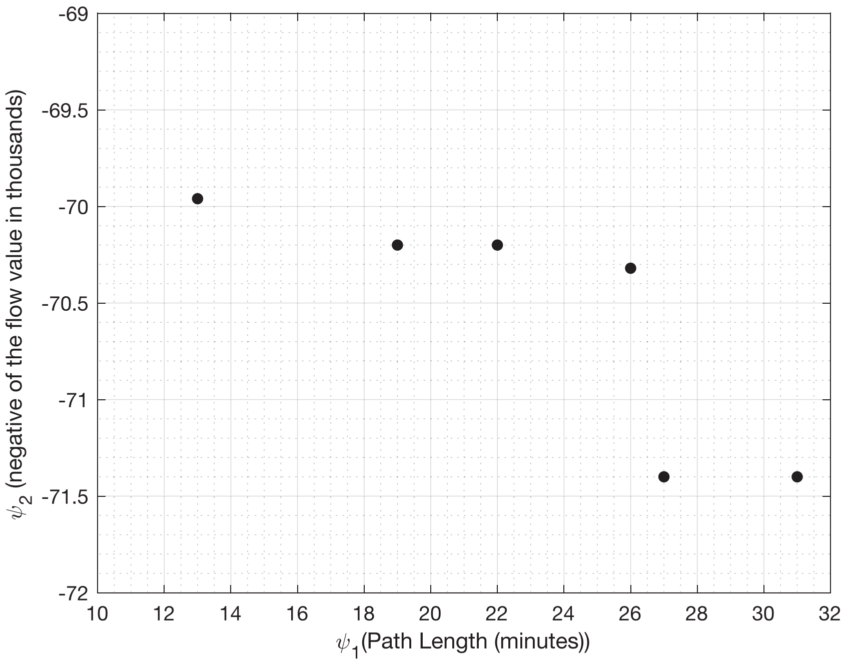


Fig. 5. Weakly non-dominated points (Kathmandu road network).

planning, we assume the capacity of each segment between 2 cars per second to 4 cars per second according to the width of the segment. The travel time to traverse the segment is taken as provided by Google Maps data. We consider node corresponding to the Tundikhel area (node 24 in Fig. 4) as the depot node d from where medical facilities, support from army, etc. can be sent.

Considering the time horizon of $T = 120$ min, the (weakly) non-dominated efficient points are depicted in Fig. 5. The non-dominated saved paths along with their path length and the corresponding maximum dynamic contraflow value are shown in Table 1.

The paths along which the flow has to be temporally repeated along with their flow values are recorded in Table 2. The arcs occupied by the flow with its direction corresponding to the saved path 24–25–26–21–20–19–18–s are shown in Fig. 6.

Using the programming language Python 3.7 on a computer with Mac operating system having 1.8 GHz dual-core Intel Core i5 processor and 8 GB RAM, the solutions are obtained in less than a second (0.85 s). The solver used to solve the mixed integer program is CBC (Coin-OR branch and cut).

Table 1. Non-dominated solution (Kathmandu road network).

Non-dominated saved path	Path length (mins.)	Flow value
24–25–26–35–15–14–13–12–s (P_1)	27	71,400
24–25–26–21–20–19–18–28–27–1–s (P_2)	26	70,320
24–25–26–21–20–19–18–28–27–s (P_3)	19	70,200
24–25–26–21–20–19–18–s (P_4)	13	69,960

Table 2. Temporally repeated flow (Kathmandu road network).

Saved path	Paths for the temporally repeated flow	Static flow per minute	Total flow value
P_1	s–1–2–38–39–40–t	240	$240 \times (120 - 37) = 19920$
	s–12–13–14–15–16–17–7–t	120	$120 \times (120 - 38) = 9840$
	s–18–28–29–30–31–32–4–5–42–41–40–t	120	$120 \times (120 - 33) = 10440$
	s–18–28–29–30–31–32–22–23–6–7–8–t	120	$120 \times (120 - 36) = 10080$
	s–27–3–4–5–6–7–8–t	120	$120 \times (120 - 37) = 9960$
	s–27–3–4–5–6–7–t	120	$120 \times (120 - 27) = 11160$
P_2	s–1–2–38–39–40–t	120	$120 \times (120 - 37) = 9960$
	s–12–13–14–15–16–17–7–t	240	$240 \times (120 - 38) = 19680$
	s–18–28–29–30–31–32–22–23–6–7–8–t	120	$120 \times (120 - 36) = 10080$
	s–18–28–29–30–31–32–4–5–6–7–8–t	120	$120 \times (120 - 35) = 10200$
	s–27–3–4–5–42–41–40–t	240	$240 \times (120 - 35) = 20400$
P_3	s–1–2–38–14–39–40–t	240	$240 \times (120 - 37) = 19920$
	s–12–13–14–15–16–17–7–t	240	$240 \times (120 - 38) = 19680$
	s–18–19–29–30–31–32–4–5–6–7–8–t	120	$120 \times (120 - 35) = 10200$
	s–18–28–29–30–31–32–4–5–6–7–8–t	120	$120 \times (120 - 35) = 10200$
	s–27–3–4–5–42–41–40–t	120	$120 \times (120 - 35) = 10200$
P_4	s–1–2–38–14–39–40–t	240	$240 \times (120 - 37) = 19920$
	s–12–13–14–15–16–17–7–t	240	$240 \times (120 - 38) = 19680$
	s–18–28–29–30–31–32–4–5–6–7–8–t	120	$120 \times (120 - 35) = 10200$
	s–27–3–4–5–6–7–8–t	120	$120 \times (120 - 37) = 9960$
	s–27–28–29–30–31–32–4–5–42–41–40–t	120	$120 \times (120 - 35) = 10200$

Hindawi
International Journal of Mathematics and Mathematical Sciences
Volume 2021, Article ID 6613622, 11 pages
https://doi.org/10.1155/2021/6613622

Research Article

Network Reconfiguration with Orientation-Dependent Transit Times

Hari Nandan Nath,[1] **Urmila Pyakurel**,[2] **and Tanka Nath Dhamala**[2]

[1] *Bhaktapur Multiple Campus, Department of Mathematics, Tribhuvan University, Bhaktapur, Nepal*
[2] *Central Department of Mathematics, Tribhuvan University, P. O. Box 13143, Kathmandu, Nepal*

Correspondence should be addressed to Urmila Pyakurel; urmilapyakurel@gmail.com

Received 5 December 2020; Revised 6 February 2021; Accepted 23 February 2021; Published 11 March 2021

Academic Editor: Fernando Bobillo

Motivated by applications in evacuation planning, we consider a problem of optimizing flow with arc reversals in which the transit time depends on the orientation of the arc. In the considered problems, the transit time on an arc may change when it is reversed, contrary to the problems considered in the existing literature. Extending the existing idea of auxiliary network construction to allow asymmetric transit time on arcs, we present strongly polynomial time algorithms for solving single-source-single-sink maximum dynamic contraflow problem and quickest contraflow problem. The results are substantiated by a computational experiment in a Kathmandu road network. An algorithm to solve the corresponding earliest arrival contraflow problem with a pseudo-polynomial-time complexity is also presented. The partial contraflow approach for the corresponding problems has also been discussed.

1. Introduction

The transportation network is represented as a dynamic network in which road segments represent the arcs and intersection between them are the nodes. The unsafe locations of network are the sources and the locations at safe regions are the sinks. Each node and arc of the network are bounded by finite capacities. Each arc has a transit time or a cost function that determines the amount of time or cost needed to travel it. The network may contain arcs in both directions with different capacities and asymmetric transit times or costs. The computational complexities for the transportation planning are heavily dependent upon the number of sources, sinks, parameters on the arcs, and nodes, like constant, time-dependent, or flow-dependent capacities or transit times as well as additional constraints. The time we consider mostly is discrete, which approximates the computationally heavy continuous models at the cost of solution approximations. Also, the constant time is probably approximated by free flow speeds or certain queuing rules and constant capacity settings that reduce the problem to be linear, at least more tractable, in contrast to the more general and realistic flow-dependent traffic flow scenarios. For extensive explanation on diversified theories and applications, we recommend Dhamala et al.[1] and Kotsireas et al. [2] and the citations therein.

The transportation network, during or after disastrous situations, becomes more congested due to large number of people and vehicles towards the safer areas on the streets. Moreover, the movement towards risk areas from safer places is discouraged because of which the corresponding road lanes are almost empty. The empty lanes management plays vital role to reduce the traffic congestion. The optimal lane reversal strategy makes the traffic systematic and smooth by removing the traffic jams caused in different large-scale natural and man-made disasters, busy office hours, special events, and street demonstrations. The contraflow reconfiguration, by means of various operations research models, heuristics, optimization, and simulation techniques, reverses the usual direction of empty lanes towards the sinks satisfying the given constraints that increase the flow value and decrease the average evacuation time [3, 4].

Although the capacity of lanes is assumed to be constant, the transit times may vary. If we consider the inflow-dependent or load-dependent transit times, the empty lanes may have likely zero time contrary to the congested segments where the transit times increase with the flow density. In reality, the transit time of incoming lanes towards the sources may not be equal to the outgoing lanes. It may be the cases depending on the network topology as well. In such cases, the free flow transit time has been considered for the reversed lanes [4]. Recently, the authors in [5] have considered the asymmetric transit times on the lanes but those lanes are individual and reversal decisions are made with their own transit times, that is, symmetric transit times such as in [3, 6, 7].

In this paper, we consider the asymmetric transit times of reversed lanes in general form and show their impact on optimal lane reversals; that is, our transit time is dependent on the orientation of the lanes which applies asymmetric times. The capacities of both reversals are the same by adding both capacities.

Consider a network in Figure 1(a). The arc labels represent the capacity of the arc and travel time on the direction of the arc. For example, the capacity of (s, a) is 4 and its travel time is 1. That means a flow of 4 units per unit time can be sent from s to a and takes 1 unit of time to reach from s to a. For a time horizon of 6 units, a maximum of 8 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, and 1 along s–a–b–d thrice) can be sent from s to d without allowing arc reversals.

If (b, a) is reversed and the transit time is kept intact (Figure 1(b)), the maximum of 10 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, 1 along s–a–b–d (that contains original (a, b)) thrice, and 2 along s–a–b–d (that contains (b, a) reversed) once) can be sent.

If the transit time depends on the orientation, when (b, a) is reversed, its transit time becomes that of original (a, b) (Figure 1(c)). In this case, there is a maximum of 14 units of flow (1 along s–a–d twice, 1 along s–b–d thrice, and 3 along s–a–b–d thrice). In this case, we can add the capacities of (a, b), (b, a) and replace the two arcs by a single arc (a, b). If (a, b) is reversed, however, the maximum flow value reduces to 5 (Figure 1(d)).

Kim et al. [8] firstly model the contraflow problem as an integer programming problem, thereby proving its $\mathcal{NP}$-hardness. As finding exact mathematical solutions for general contraflow techniques is costly, they present two greedy and bottleneck heuristics for possible numerical approximate solutions to the quickest contraflow problem. With computational experiments, it has been shown that at least 40% evacuation time can be reduced by reverting at most 30% arcs. Rebennack et al. [3] solve the two-terminal maximum and quickest contraflow problems optimally in strongly polynomial times. Pyakurel and Dhamala [6] solve the earliest arrival contraflow problem on a two-terminal network in pseudo-polynomial-time. They solve it in strongly polynomial time if the network is series-parallel. The continuous time solution is given in [7] and the solutions with similar objectives are given in [9, 10]. They show that if the minimum cut arcs have symmetric capacities, then the flow is double with the contraflow. Pyakurel et al. [11] give the first temporally repeated flow algorithm to solve the quickest contraflow problem, within a complexity of solving a min-cost flow problem. The costs for arc reversals as switching costs are studied in [12]. The contraflow models with intermediate storage are introduced in [13].

In this paper, we introduce the maximum dynamic contraflow, earliest arrival, and quickest contraflow problems on asymmetric transit time network and present efficient algorithms to solve these problems in two-terminal networks. Modifying the network transformation suggested by Rebennack et al. [3] in case of symmetric travel time cases, we show that the approach works equally well in the asymmetric travel time settings. The results are extended with partial lane reversals as well. We analyze our solutions in both discrete and continuous time settings. The novelty of this work is to optimize network topology with unequal transit times on reversal arcs to improve congestion by increasing the flow value and decreasing the evacuation time.

We organize the paper as follows. Section 2 presents mathematical formulations and models of the problems allowing asymmetric transit times on arcs. We investigate the maximum dynamic, earliest arrival, and quickest flow problems in Section 3. These results are extended with partial lane reversals in Section 4. In Section 5, some computational results, taking a part of Kathmandu road network as an example network, are presented. The paper is concluded in Section 6.

2. Basic Concepts

A network N is a directed graph consisting of a finite set of nodes V and a finite set of arcs A with $|V| = n$, $|A| = m$. An arc $e \in A$ is associated with a unique pair of nodes i, j: one of them is called the head of e and the other the tail. If e has head i and tail j, then it is called directed from i to j. We consider something known as flow that moves from a set of nodes $S \subset V$, called sources to $D \subset V$ $(S \neq D)$. The amount of flow is limited by a capacity function $u\colon A \longrightarrow \mathbb{R}_{\geq 0}$. A travel time $\tau\colon A \longrightarrow \mathbb{R}_{\geq 0}$ is associated with the flow. We denote such a network by $N = (V, A, u, \tau, S, D)$. If $S = \{s\}$ and $D = \{d\}$, we denote the network by $N = (V, A, u, \tau, s, d)$. We define the set of arcs incoming to node i as

$$A_i^- = \{e \in A\colon i \text{ is the head of } e\}, \tag{1}$$

and the set of arcs going out of it as

$$A_i^+ = \{e \in A\colon i \text{ is the tail of } e\}. \tag{2}$$

2.1. Static Flow. A static flow is a function $x\colon A \longrightarrow \mathbb{R}_{\geq 0}$ which satisfies the following conditions:

$$\sum_{e \in A_i^+} x(e) - \sum_{e \in A_i^-} x(e) = 0 \quad \forall i \in V \setminus \{S, D\}, \tag{3}$$

$$x(e) \leq u(e) \quad \forall e \in A. \tag{4}$$

The value of the static flow x is

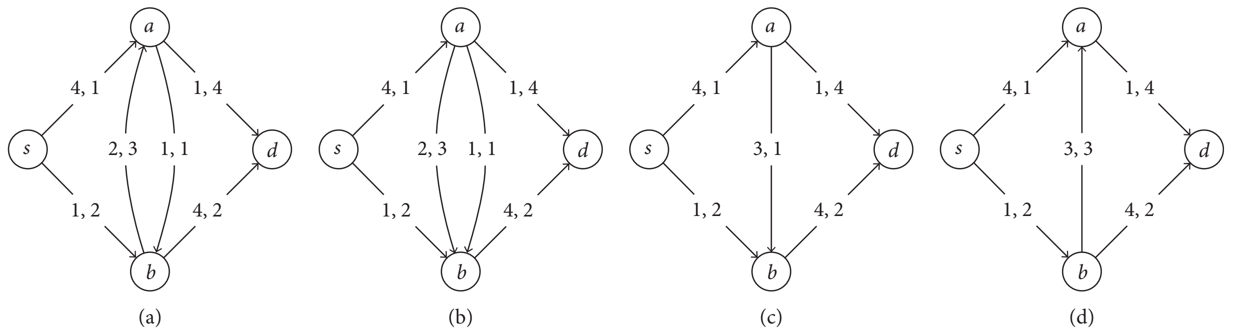


Figure 1: (a) Network with arc labels capacity and transit time; (b) (b, a) reversed with the same transit time; (c) (b, a) reversed with orientation-dependent transit time; (d) (a, b) reversed with orientation-dependent transit time.

$$v(x) = \sum_{d\in D}\sum_{e\in A_d^-} x(e). \tag{5}$$

A static flow x is called a circulation if (3) is satisfied by all $i \in V$. A flow x that maximizes $v(x)$ is called a maximum static flow.

2.1.1. Flow Decomposition.

2.1.1. Flow Decomposition. Let $\mathscr{P}$ denote the set of all simple paths from S to D, and let C denote the set of all simple cycles in N. Then every static flow x has a flow decomposition $(x_P)_{P\in\mathscr{P}\cup\mathscr{C}}$ such that $x_P \geq 0\,\forall P \in \mathscr{P}\cup\mathscr{C}$ and

$$\sum_{P\in\mathscr{P}\cup\mathscr{C}x_P:\ e\in P} x_P = x(e). \tag{6}$$

2.1.2. Minimum Cost Flow. Given two functions $b: V \longrightarrow \mathbb{R}$, called supply, and $c: A \longrightarrow \mathbb{R}$, called cost with $\sum_{i\in V} b(i) = 0$, a flow x, satisfying (4) and

$$\sum_{e\in A_i^+} x(e) - \sum_{e\in A_i^-} x(e) = b(i) \quad \forall i \in V, \tag{7}$$

is called a minimum cost flow if it minimizes $\sum_{e\in A} c(e)x(e)$. In various applications, the travel time τ is considered as cost.

2.1.3. Residual Network. A very important notion for various network flow calculations is a residual network. Given a static flow x, the residual network N_x has the same vertex set V. The arc set A_x consists of arcs constructed in the following way: For each $e \in A$ directed from i to j, if $x(e) < u(e)$, there is an arc in A_x directed from i to j with residual capacity $u(e) - x(e)$ and cost $c(e)$. If $x(e) > 0$, we have an arc in A_x directed from j to i with residual capacity $x(e)$ and cost $-c(e)$.

For more details, we refer the reader to Ahuja et al. [14].

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

$$\operatorname{exc}_\Phi(i, \theta) = \sum_{e\in A_i^-}\int_0^{\theta-\tau(e)} \Phi_e(\sigma)\mathrm{d}\sigma - \sum_{e\in A_i^+}\int_0^{\theta} \Phi_e(\sigma)\mathrm{d}\sigma, \tag{8}$$

which is the net amount of flow that is stored at node i up to time θ. In what follows, we assume $S = \{s\}, D = \{d\}$ for simplicity. A feasible dynamic flow Φ satisfies

$$\operatorname{exc}_\Phi(i, \theta) \geq 0 \quad \forall\theta \in [0, T),\ \forall i \in V\setminus\{s\}, \tag{9}$$

$$\operatorname{exc}_\Phi(i, T) = 0, \quad \forall i \in V\setminus\{s, d\}, \tag{10}$$

$$0 \leq \Phi_e(\theta) \leq u(e), \quad \forall e \in A, \theta \in [0, T). \tag{11}$$

The value of the dynamic flow Φ at time θ is

$$v_\theta(\Phi) = \operatorname{exc}_\Phi(d, \theta), \tag{12}$$

and the total value of the dynamic flow Φ is

$$v(\Phi) = v_T(\Phi) = \operatorname{exc}_\Phi(d, T). \tag{13}$$

For more details, refer to Skutella [15].

Given a time horizon T, the dynamic flow Φ that maximizes $v_T(\Phi)$ is called the *maximum dynamic flow*. Given a flow value Q, the dynamic flow with minimum time horizon T^* such that $v_{T^*}(\Phi) = Q$ is called the *quickest flow*, and the dynamic flow Φ which maximizes $v_\theta(\Phi)$ for all $\theta \in [0, T]$ is called the *earliest arrival flow*.

2.2.1. Temporally Repeated Flow. Given a static flow x and a time horizon T, a flow decomposition on x gives a set of paths $\mathscr{P}$ with flow x_P for each $P \in \mathscr{P}$. Flow is sent along P at a constant rate x_P from time 0 to $\max\{T - \tau(P), 0\}$, where $\tau(P) = \sum_{e\in P}\tau(e)$ is the travel time on path P, to define a dynamic flow known as the temporally repeated flow. To give an explicit expression for the dynamic flow, we define the following for e in a path P directed from i to j:

P_{si} = the portion of the path P from s to i,

$P_{j\,d}$ = the portion of the path P from j to d,

$$\mathscr{P}_e(\theta) = \left\{P \in \mathscr{P}\colon e \in P, \tau(P_{si}) \leq \theta, \tau\left(P_{j\,d}\right) < T - \theta\right\}. \tag{14}$$

Now, the dynamic flow Φ is defined by

$$\Phi_e(\theta) = \sum_{P \in \mathcal{P}_e(\theta)} x_P \quad \forall e \in A, \theta \in [0, T). \tag{15}$$

2.2.2. Discrete Dynamic Flow. Discretizing the time intervals $[0, T)$ into the time steps $0, 1, \ldots, T-1$, each corresponding to $[0, 1), [1, 2), \cdots [T-1, T)$, we can replace the integral sign in (9) by a summation sign (removing $d\sigma$); the corresponding flow is known as discrete dynamic flow. Using the concept of natural transformations, Fleischer and Tardos [16] show the equivalence between the two problems so that the solution procedures of a problem in continuous time version can be carried to the solution procedure of the corresponding problem in the discrete version, and vice versa.

3. Dynamic Contraflow Solutions

We consider the network N with set of nodes V, set of arcs A, capacity $u: A \longrightarrow \mathbb{R}_{\geq 0}$, and travel time functions $\tau, \overleftarrow{\tau}: A \longrightarrow \mathbb{R}_{\geq 0}$. For each $e \in A$, with the tail node i and head node j, $\tau(e)$ or τ_e denotes the arc transit time from i to j and $\overleftarrow{\tau}(e)$ or $\overleftarrow{\tau}_e$ denotes arc transit time from j to i. Without loss of generality, we make the following conventions:

(1) There exist at most two arcs (with opposite orientations) between any two nodes i and j. We denote the arc $e$with the tail i and the head j by (i, j).

(2) Whenever $(i, j) \in A$, there is $(j, i) \in A$. This can be done, for our purpose, by assigning $u_{ji} = 0$ if such an arc does not exist.

(3) Defining $\tau(j, i) = \overleftarrow{\tau}(i, j)$, we assume the existence of only one travel time function τ.

Remark 1. In case the above-mentioned conventions are not satisfied, we can use suitable network transformations to meet the requirements. For example, if $\tau(i, j) \neq \overleftarrow{\tau}(i, j)$, we can transform the network N_0 in Figure 2(a) to the network N shown in Figure 2(b). Given the capacities $u(i, j) = u_{ij}, u(j, i) = u_{ji}$ and transit times $\tau(i, j) = \alpha_1, \overleftarrow{\tau}(i, j) = \alpha_2, \tau(j, i) = \beta_1, \overleftarrow{\tau}(j, i) = \beta_2$, we replace arc (i, j) in N_0 by two arcs $(i, j), (j, i)$ with capacities and transit times u_{ij}, α_1 and $0, \alpha_2$, respectively. The arc (j, i) in N_0 is replaced by $(i, k), (k, j), (j, k), (k, i)$ adding a node k to the network to avoid parallel arcs. The capacities and travel times of $(j, k), (k, i)$ are taken as $u_{ji}, 0$ and u_{ji}, β_1, respectively, and those of $(i, k), (k, j)$ are taken as $0, \beta_2$, and $0, 0$, respectively.

Definition 1 (Auxiliary network). For each $N = (V, A, u, \tau, s, d)$, we define the auxiliary network as $N' = (V', A', u', \tau', s, d)$, in which

(1) $V' = V, A' = A$

(2) $\forall (i, j) \in A, u'(i, j) = u(i, j) + u(j, i)$

(3) $\forall (i, j) \in A', \tau'(i, j) = \tau(i, j)$

Example 1. Consider a network N as depicted in Figure 3(a). The arc labels represent the capacity and the transit time. The auxiliary network N' of N is constructed in Figure 3(b). The capacity of each arc is the sum of its capacity and the opposite arc and the transit time is the same as that of the corresponding arc in N.

3.1. Maximum Dynamic Contraflow

Problem 1 (maximum dynamic contraflow problem with orientation-dependent transit times). Given a network $N = (V, A, u, \tau, s, d)$ with transit time τ depending on the orientation and a time horizon T, find the maximum dynamic flow allowing the arc reversals at time 0.

According to Ford and Fulkerson [17], the problem of finding the static flow corresponding to the temporally repeated maximum dynamic flow can be formulated as

$$\min -Tv + \sum_{(i,j)\in A} \tau(i, j) x(i, j),$$
$$\sum_{e\in A_i^+} x(i, j) - \sum_{(i,j)\in A_i^-} x(i, j) = \begin{cases} v, & i = s \\ -v, & i = d \\ 0, & i \in V \backslash \{s, d\} \end{cases},$$
$$0 \leq x_{(i,j)} \leq u(i, j) \forall (i, j) \in A. \tag{16}$$

Problem 1 is to find the maximum dynamic flow so that an arc (i, j) can take also the capacity of (j, i) and vice versa. So, $x(i, j) + x(j, i))$ must not exceed $u(i, j) + u(j, i)$. However, the removal of cycle flows does not change the value of the static flow v and does not improve $-Tv + \sum_{(i,j)\in A} \tau(i, j) x(i, j)$; we can impose the condition that either $x(i, j) = 0$ or $x(j, i) = 0$. So, the problem to find the static flow corresponding temporally repeated maximum dynamic flow can be stated as

$$\min -Tv + \sum_{(i,j)\in A} \tau(i, j) x(i, j), \tag{17}$$

$$\sum_{e\in A_i^+} x(i, j) - \sum_{(i,j)\in A_i^-} x(i, j) = \begin{cases} v, & i = s, \\ -v, & i = d, \\ 0, & i \in V \backslash \{s, d\}, \end{cases} \tag{18}$$

$$0 \leq x_{(i,j)} \leq u(i, j) + u(j, i) \forall (i, j) \in A, \tag{19}$$

$$x(i, j) \cdot x(j, i) = 0. \tag{20}$$

See also [17] for the similar formulation of maximizing the static flow in undirected and mixed networks. The problem in (17)–(19) is a linear programming problem and (20) can be satisfied by the removal of cycle flows in the solution. Since a linear programming problem is polynomial solvable and using flow decomposition, removal of cycle flows also can be done in polynomial time (see [14]); we can find the static flow corresponding to the temporally repeated maximum dynamic flow allowing arc reversals in a

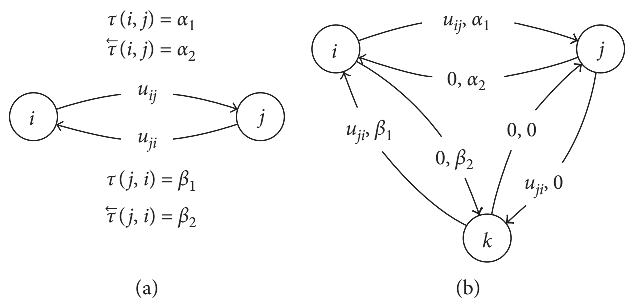


FIGURE 2: Network transformation for the cases $\tau(j,i) \neq \overleftarrow{\tau}(i,j)$. (a) Network N0. (b) Network N.

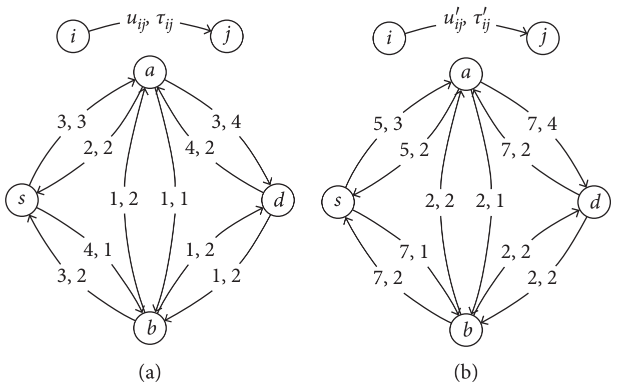


FIGURE 3: (a) Evacuation network N. (b) Auxiliary network N'.

polynomial time. If the static flow in an arc exceeds its capacity, the opposite arc has to be reversed at time zero.

In Algorithm 1, we present a procudure to solve Problem 1.

Theorem 1. *Algorithm 1 solves the maximum dynamic contraflow problem with orientation-dependent transit times correctly.*

Proof. It is easy to see that steps 1–3 are well defined. Step 4 may be ill defined if for some $(i,j) \in A', x(i,j), x(j,i) > 0$. But because the removal of cycle flows in Step 3 ensures either $x(i,j)$ or $x(j,i)$ to be zero, Step 4 is also well defined. Moreover, $x(i,j) \leq u'(i,j) = u(i,j) + u(j,i)$. This shows that x is feasible in N after necessary arc reversals.

Since the cycle flows do not contribute to the value of the flow, x is optimal in N' even after removing the cycle flows in Step 3. Let v^x be the value of such a dynamic flow. We claim that x is optimal in N (after arc reversals). If not so, there exists an instance of arc reversals, N^y, of N in which we can find a static flow y, temporal repetition of which results in a dynamic flow with flow value more than v^x. In N^y, we can replace (i,j) and (j,i) by a single arc (i,j) with transit time $\tau(i,j)$ and capacity $u(i,j) + u(j,i)$ if (i,j) has been reversed. Let $\mathcal{P}$ be the set of paths corresponding to the path decomposition of y. Corresponding to each path P: s–i_1–i_2–$\cdots$–i_k–d in $\mathcal{P}$, we have a path P': s–i_1–i_2–$\cdots$–i_k–d in N'. Each arc in P' has the same transit time and the capacity not less than that of the corresponding arc in P. Let the collection of such paths P' be $\mathcal{P}'$. Defining a flow $y'(i,j) = y(i,j)$ for each $(i,j) \in P'$, we can find a dynamic flow in N' with a flow value more than v^x. This contradicts the optimality of x in N' also.

Hence, Algorithm 1 computes the maximum dynamic flow in N reversing appropriate arcs. □

Theorem 2. *Algorithm 1 runs in $O(MDF + nm)$ time where $n = |V|, m = |A|$ and MDF is the running time of a temporally repeated maximum dynamic flow computation.*

Proof. The construction of the auxiliary network in Step 1 requires $O(m)$ time. The flow decomposition in Step 3 can be done in $O(mn)$ time (Ahuja et al.) [14]. Step 4 can be performed in $O(m)$ time. Thus, if MDF is the running time of Step 2, the algorithm runs in $O(\text{MDF} + nm)$ time. However, in general, this complexity could be considered as the complexity of MDF solution. □ □

Theorem 3. *The maximum dynamic contraflow problem with orientation-dependent transit times can be solved in a strongly polynomial time.*

Proof. From Theorem 2, the running time of Algorithm 1 is $O(\text{MDF} + nm)$, where MDF is the running time of a maximum dynamic flow calculation. In $N = (V, A, u, \tau, s, d)$, and consequently in the auxiliary network N', we can find the static flow x corresponding to the maximum dynamic flow as follows [16]:

```
    Input: A network $N = (V, A, u, \tau, s, d)$ with orientation dependent transit time $\tau$, and a time horizon $T$
    Output: Maximum dynamic flow with arc reversals
(1) Construct the auxiliary network $N' = (V', A', u', \tau', s, d)$ according to Definition 1.
(2) Find a static flow $x$ corresponding to the temporally repeated dynamic flow in $N'$.
(3) Decompose $x$ into chain-flows and cycle-flows. Remove the cycle-flows and update $x$.
(4) Reverse $(j, i) \in A$ if and only if $x(i, j) > u(i, j)$. Dynamic flow in reconfigured $N$ = a temporally repeated flow corresponding to $x$
    with the time horizon $T$.
```

Algorithm 1: Maximum dynamic contraflow with orientation-dependent transit times.

(1) Add an arc (d, s) with $u(d, s) = \infty$ and $\tau'(d, s) = -T$ to the auxiliary network N' to obtain a network N''.

(2) Calculate the minimum cost circulation x' in N''.

(3) x = the restriction of x' to N'.

Using the enhanced capacity scaling algorithm mentioned by Orlin [18], the minimum cost circulation in N'' can be calculated in $O(m \log n(m + n \log n))$ time. This proves the assertion. □ □

3.2. Quickest Contraflow

Problem 2. (Quickest contraflow problem with orientation-dependent transit times). Given a network $N = (V, A, u, \tau, s, d)$ with transit time τ depending on the orientation and a supply Q at s, find the dynamic flow with minimum time horizon allowing the arc reversals at time 0.

We construct Algorithm 2 to solve Problem 2.

Let $v(T)$ be the value of the maximum dynamic flow with time horizon T. One of the strategies to find the quickest flow with a supply Q at s is to search for the minimum time horizon T^* such that $Q \leq v(T^*)$. Various search strategies are described in Burkard et al.'s work [19]. This requires solving the maximum dynamic flow problem repeatedly. Using this technique in Step 2, the feasibility and optimality arguments given in Theorem 1 are valid in Algorithm 2 as well. Hence, we have the following result.

Theorem 4. *Algorithm 2 solves the quickest contraflow problem with orientation-dependent transit times correctly.*

Analogous to the case of solving the maximum dynamic contraflow problem, if QF is the complexity of solving the quickest flow problem in N, we have the following.

Theorem 5. *Algorithm 2 runs in $O(QF + nm)$ time, where QF is the running time of a temporally repeated quickest flow problem.*

Since there exist strongly polynomial algorithms for solving the quickest flow problem, we have the following.

Theorem 6. *The quickest contraflow problem with orientation-dependent transit times can be solved in a strongly polynomial time.*

Proof. Consider a network $N = (V, A, u, \tau, s, d)$ with a supply Q at the source s. We show that the complexity of Algorithm 2 is strongly polynomial to reach the conclusion. Using the cancel-and-tighten algorithm described in the work of Saho and Shigeno [20], to solve the quickest flow problem Step 2 takes $O(nm^2 \log^2 n)$ time. Using the fact in Theorem 5, the complexity of Algorithm 2 is $O(nm^2 \log^2 n)$. This proves the assertion. □ □

3.3. Earliest Arrival Contraflow. Consider a network $N = (V, A, u, \tau, s, d)$ with orientation-dependent transit time τ. With a given time horizon T, the earliest arrival flow problem seeks to find the flow that is maximum with each time horizon $\theta \in [0, T]$. Carrying over the idea of solving the maximum dynamic contraflow problem, we try to solve the earliest arrival flow problem on the auxiliary network N'. This can be carried out by implementing the successive shortest path algorithm in the residual network of N' ([21, 22]). The idea is to send the flow successively through shortest paths amounting the residual capacity of the path. Such a path, being in the residual network, may contain negative (cost) time arcs also. Sending flow through negative cost arcs sends the flow back in time. This may result in the arc used by the flow in some time interval, not used in the other time interval, and used again in some further time interval. So, an arc reversed to optimize the flow at one time does not remain reversed throughout the given whole-time interval. In other words, an arc may have to be flipped back and forth unlike in the cases of maximum dynamic contraflow and quickest contraflow, where an arc reversed at the beginning (time zero) remains reversed throughout.

Problem 3. Given a network $N = (V, A, u, \tau, s, d)$ with orientation-dependent transit time τ and a time horizon T, find a dynamic flow which is maximum at every time horizon θ such that $0 \leq \theta \leq T$ allowing arc reversals at time θ.

We propose Algorithm 3, which sends the flow along the successive shortest paths in the residual network of the auxiliary network.

The earliest arrival flow in Step 2 can be found by sending the flow equal to the residual capacity along the shortest paths in the residual network of N'. The generalized temporally repeated flow can be found in [15]. Such an algorithm runs in a pseudo-polynomial-time.

```
Input: A network $N = (V, A, u, \tau, s, d)$ with orientation dependent transit time $\tau$, and a supply $Q$ at $s$
Output: Quickest flow with the arc reversals
(1) Construct the auxiliary network $N' = (V', A', u', \tau', s, d)$ according to Definition 1.
(2) Find a static flow $x$ corresponding to the temporally repeated quickest flow in $N'$, and the quickest time $T^*$.
(3) Decompose $x$ into chain-flows and cycle-flows. Remove the cycle-flows and update $x$.
(4) Reverse $(j, i) \in A$ if and only if $x(i, j) > u(i, j)$. Quickest flow in reconfigured $N$ = a temporally repeated flow corresponding to $x$
    with the time horizon $T^*$.
```

Algorithm 2: Quickest contraflow with orientation-dependent transit times.

```
Input: A network $N = (V, A, u, \tau, s, d)$ with orientation dependent transit time $\tau$, and a time horizon $T$
Output: Earliest arrival flow with the arc reversals
(1) Construct the auxiliary network $N' = (V', A', u', \tau', s, d)$ according to Definition 1.
(2) Find the earliest arrival flow $f$ in $N'$.
(3) Reverse $(j, i) \in A$ if $f_{ij}(\theta) > u(i, j)$ at time $\theta \in [0, T]$. $f$ is the earliest arrival flow with arc reversals in $N$.
```

Algorithm 3: Earliest arrival contraflow with orientation-dependent transit times.

Theorem 7. *Algorithm 3 solves the earliest arrival contraflow problem with orientation-dependent transit times in a pseudo-polynomial-time.*

However, in a series-parallel network with a single source and single sink, the maximum dynamic flow has the earliest arrival property (Ruzika et al.) [23]. So, in case of a series-parallel network N, Algorithm 1 solves Problem 3. Moreover, the maximum dynamic flow in the auxiliary network of a series parallel network with time horizon T can be solved in $O(mn + m \log m)$ time by sending the flow iteratively through the s–d paths with the minimum time and removing the saturated arcs, considering only the paths with time not exceeding T. So, we have the following.

Theorem 8. *If $N = (V, A, u, \tau, s, d)$ is a series-parallel network, then the earliest arrival contraflow problem with orientation-dependent transit times can be solved in $O(mn + m \log m)$ time.*

4. Partial Contraflow Algorithm

The approach described in Section 3 either reverses an arc or does not reverse it. In various applications, for example, evacuation planning, an arc refers to a collection of lanes in a particular direction; it is beneficial if we reverse the lanes required by the flow. The unused lanes may be used for other facilities [24]. Realizing the need of arc reversals up to the required capacity only, the authors in [4] introduce the concept of partial arc reversals. We extend the procedure in case of orientation-dependent transit times on arcs.

Algorithm 4 describes the generic procedure to solve the corresponding problems described in Section 3. The algorithm not only reverses the arcs up to the necessary capacity but also lists the unused capacities of the arcs of the network considered.

The correctness of Algorithm 4 can easily be realized from the correctness of the corresponding algorithm with full arc reversal and the following fact. For the arcs (i, j), (j, i) between nodes i and j, it is evident that either only one of them is reversed or both of them are not reversed. If (i, j) is reversed, it clearly indicates that $x(j, i) > u(j, i)$; there is no capacity of (j, i) unused; that is, $r(j, i) = 0$, and

$$\begin{aligned} r(i, j) &= u(i, j) - [x(j, i) - u(j, i)] \\ &= u(i, j) + u(j, i) - x(j, i) \\ &= u'(i, j) - x(j, i). \end{aligned} \tag{21}$$

If both arcs are not reversed, then $x_{ij} \le u(i, j)$ meaning $r(i, j) = u(i, j) - x(i, j)$.

The flow x in Step 1 of Algorithm 4 has to be considered as the corresponding static flow in case of the maximum dynamic contraflow (Section 3.1) and quickest flow (Section 3.2) and the dynamic flow in case of the earliest arrival flow (Section 3.3). Step 2 has to be implemented at time zero and at time θ accordingly.

As the extra procedure of listing the unused capacities in Step 3 takes only $O(m)$ time, the overall complexities of the algorithms in case of partial contraflow are the same as those of the contraflow. Thus, we have the following.

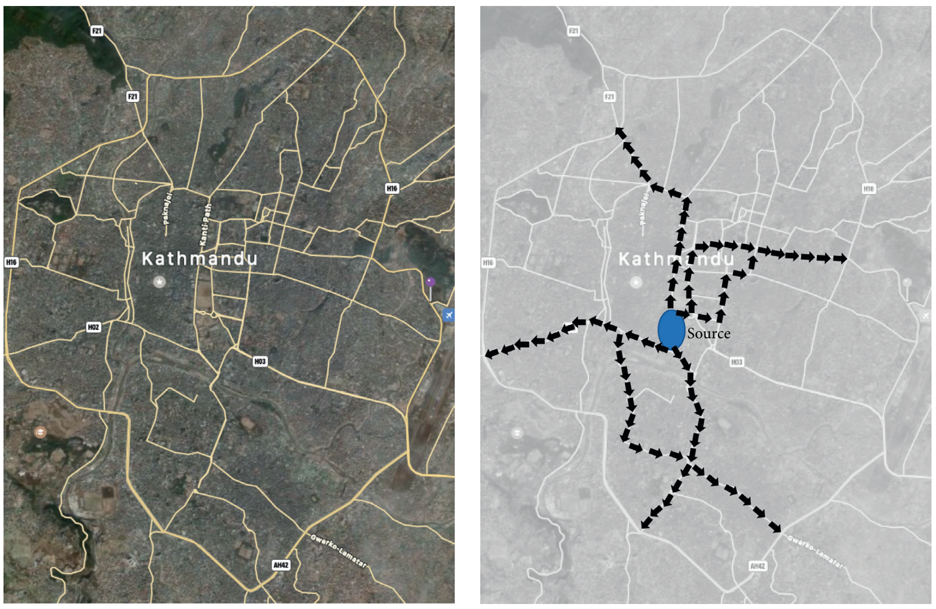
Theorem 9. *The worst-case complexity of a partial contraflow problem with orientation-dependent transit times is the same as that of the corresponding contraflow problem.*

5. Computational Experiment

For testing the computational performance of the maximum dynamic contraflow algorithm (Algorithm 1) and the quickest contraflow algorithm (Algorithm 2), we consider Kathmandu road network inside Ring Road with major road segments as an example network (Figure 4). We consider a

```
    Input: Network $N = (V, A, u, \tau, s, d)$ with orientation dependent transit times
    Output: Partial contraflow reconfiguration of $N$ with unused capacities $r$.
(1) Find a cycle-free flow $x$ in the auxiliary network $N' = (V', A', u', \tau', s, t)$.
(2) Reverse $(j, i) \in A$ proportional to the capacity $x(i, j) - u(i, j)$ if $x(i, j) > u(i, j)$.
(3) For each $(i, j) \in E$, if $(i, j)$ is reversed, then $r_{ij} = u'_{ij} - x_{ji}$ and $r_{ji} = 0$. If neither $(i, j)$ nor $(j, i)$ is reversed, $r(i, j) = u(i, j) - x(i, j)$.
```

Algorithm 4: Generic partial contraflow algorithm.



(a) (b)

Figure 4: (a) Kathmandu road network. (b) Maximum flow direction with arc reversals $T = 1$ hour.

Table 1: Network data for computations (Section 5).

i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}	i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}	i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}
0	49	3	3	85	70	0	1	4	0	70	70	0	16	0	4	36	36
0	19	2	2	41	41	1	2	4	0	18	16	1	14	2	2	18	18
2	3	2	2	47	47	2	12	4	0	21	21	3	4	2	2	32	25
3	11	2	2	21	21	4	5	2	2	10	25	4	56	2	2	35	50
5	6	2	2	180	150	5	9	2	2	65	70	6	7	2	2	90	110
7	8	2	2	85	95	7	33	2	2	45	55	8	9	2	2	30	25
8	30	2	2	25	30	8	32	2	2	24	30	9	10	2	2	35	45
10	11	2	2	34	34	10	27	2	2	30	35	10	30	2	2	64	64
11	12	2	2	30	35	12	13	4	0	20	25	12	27	2	2	40	40
13	14	4	0	26	26	14	15	4	0	40	42	14	25	2	2	55	60
15	16	4	0	24	27	15	24	2	2	50	50	16	22	0	4	52	52
16	23	2	2	41	41	19	20	2	2	50	50	19	22	2	2	45	40
20	21	2	2	100	105	20	48	2	2	130	145	21	47	2	2	50	50
21	48	2	2	153	153	22	23	2	2	36	40	22	45	4	4	145	145
23	24	2	2	34	30	24	25	2	2	42	38	25	26	2	2	30	25
25	42	0	3	36	28	26	27	2	2	18	16	27	28	2	2	32	32
28	29	2	2	34	30	28	41	2	2	17	15	29	30	2	2	18	16
29	40	2	2	24	24	30	31	2	2	24	28	31	32	2	2	25	22
32	33	2	2	22	20	33	34	2	2	20	25	34	35	2	2	32	30

Table 1: Continued.

i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}	i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}	i	j	u_{ij}	u_{ji}	τ_{ij}	τ_{ji}
34	37	2	2	88	80	35	37	2	2	24	28	36	37	2	2	35	40
38	39	2	2	65	75	39	40	2	2	47	47	39	43	2	2	32	35
40	41	2	2	30	34	41	42	2	2	30	34	41	43	2	2	37	37
42	44	2	2	120	135	44	45	2	2	135	135	45	46	2	2	135	135
48	49	2	2	126	126	49	50	2	2	38	38	50	51	2	2	38	38
50	64	2	2	20	20	51	52	2	2	80	75	51	63	2	2	130	125
51	64	2	2	45	45	52	53	2	2	50	45	52	61	2	2	40	35
53	54	2	2	18	18	53	61	2	2	25	22	54	55	2	2	68	68
54	58	2	2	70	70	55	56	2	2	36	34	55	57	2	2	76	70
55	58	2	2	82	75	57	58	2	2	80	75	58	59	2	2	45	40
59	60	2	2	60	65	60	62	2	2	34	34	62	63	2	2	68	68
64	101	2	2	180	160	101	999	2	2	0	0	63	999	2	2	60	60
62	999	2	2	108	100	60	999	2	2	117	110	59	104	2	2	47	47
104	999	2	2	0	0	59	999	2	2	45	45	57	999	2	2	21	21
56	999	2	2	171	171	6	999	2	2	108	108	35	999	2	2	153	153
36	110	2	2	90	100	110	999	2	2	0	0	36	999	2	2	70	65
38	112	2	2	50	40	112	999	2	2	0	0	38	999	2	2	41	36
43	999	2	2	100	90	44	999	2	2	145	135	46	999	4	4	50	50
21	999	2	2	190	200	47	118	2	2	198	198	118	999	2	2	0	0
47	999	2	2	80	80	64	999	2	2	135	135						

Source node: 0; sink node: 999.

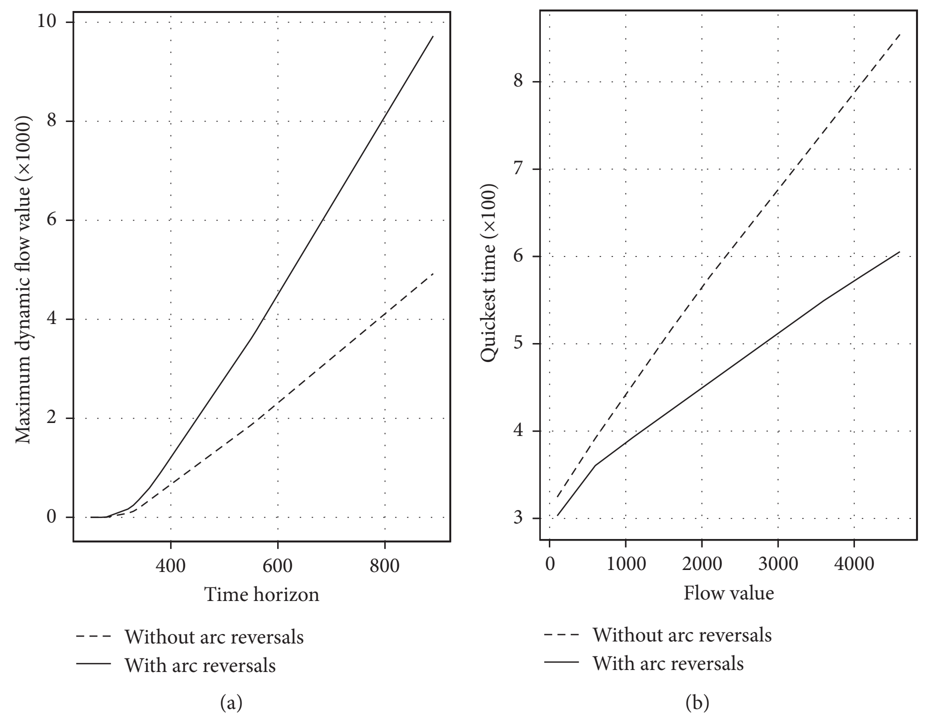


Figure 5: Flow value and time comparisons.

problem of evacuating people in Dasharath Stadium and surrounding area to outside of the Ring Road in case of auto-based-emergency evacuation.

The constructed directed network consists of 69 nodes and 232 arcs (see Table 1). The direction of arcs is taken as the direction of the usual flow of the traffic in the

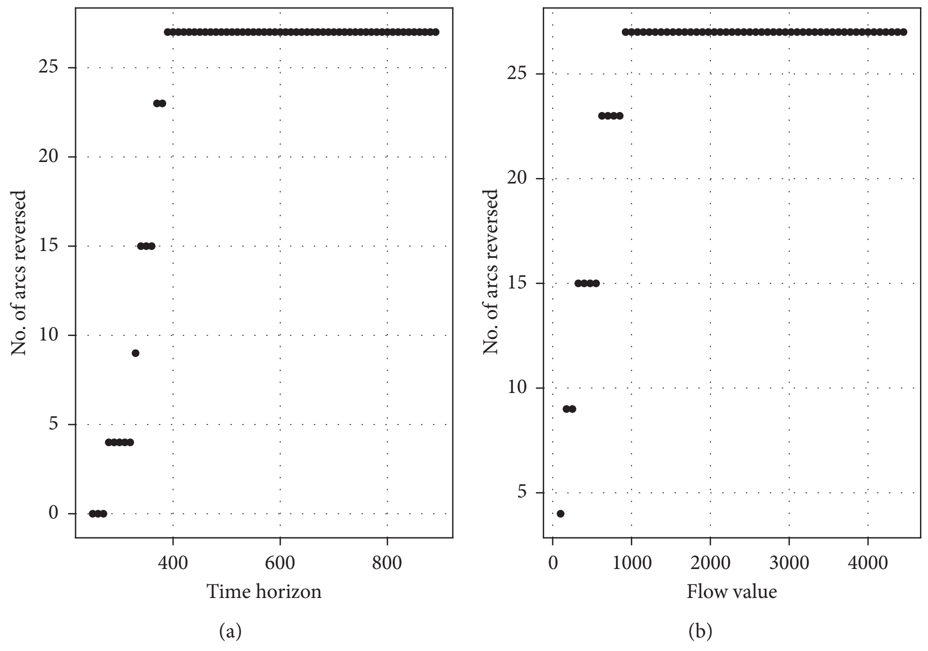


Figure 6: Number of arcs reversed.

corresponding segments. The capacities of the road segments are taken from 2 to 4 units of flow per second according to the number of lanes. The travel time (in seconds) in one of the arcs between any two nodes is considered according to the length and that in the opposite arc is chosen differing from it by 0 to 30 seconds (chosen randomly). The considered data are taken only for the purpose of testing the algorithms. The accuracy of capacity and travel time demands complex technical examinations.

We calculate the maximum dynamic flow with and without contraflow taking time horizons from as low as 5 minutes to as high as 1 hour. At each time horizon considered, we find that the value of the flow after allowing arc reversals is almost the double of that without arc reversals (Figure 5(a)). At $T = 5$ minutes, it is 44 without arc reversals and 88 with arc reversals. At $T = 60$ minutes, the corresponding values are 29,312 and 58,502.

Given a flow value at the source, the calculation of quickest flow shows that the decrease in the quickest time increases with the increase in the flow value. With the flow value as low as 500, the quickest time decreases only by 7%, which is 47% for the flow value 50,000.

With growing time horizon and growing flow value, as well as the number of arcs, the flow occupies more and more arcs and, consequently, the number of arcs reversed increases. However, it remains fixed after some value of time or the flow value (Figure 6(a) for maximum dynamic flows calculations and Figure 6(b) for quickest flows).

Among the considered instances, the running time of a maximum contraflow calculation is at most 0.013 seconds, and that of a quickest contraflow calculation is at most 0.067 seconds. The coding is done in Python programming language and run in MacOS 11.1 with 1.8 GHz Dual-Core Intel Core i5 processor, and 8 GB RAM.

6. Conclusion

In this work, we introduce the contraflow problem in which the transit time on an arc depends on the direction of the arc; that is, the transit time on an arc may change after its reversal. This extends the notion, in the existing literature, that the transit time on arcs remains the same before and after the arc reversal to the cases where the time on arcs depends on its orientation. Presenting a method of constructing an auxiliary network, strongly polynomial time algorithms for maximum dynamic contraflow problem and quickest contraflow problem with orientation-dependent transit times are presented for a single-source-single-sink network. In the similar settings, for the earliest arrival contraflow problem, a pseudo-polynomial-time algorithm is also presented. The computational performance of the algorithms for maximum dynamic contraflow and quickest contraflow taking a Kathmandu road network is also tested.

The presented approach is useful, particularly, in transportation planning, where the transit time depends on the direction of the traffic flow because of various reasons, for example, topography of the road. When the direction of the traffic flow in a road segment is reversed, if the capacity permits, it is beneficial to reverse only the necessary lanes. To address such an issue, we present corresponding algorithms in the partial contraflow setting as well. Analyzing impressive results from this research, its further extensions to flow-dependent scenarios would be interesting problems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The first author acknowledges the support of UGC Nepal for Ph.D. fellowship, and the second author acknowledges the support of AvH Foundation for the return fellowship.

References

[1] T. N. Dhamala, U. Pyakurel, and S. Dempe, "A critical survey on the network optimization algorithms for evacuation planning problems," *International Journal of Operations Research*, vol. 15, pp. 101–133, 2018.

[2] I. S. Kotsireas, A. Nagurney, and P. M. Pardalos, *Dynamics of Disasters-Algorithmic Approaches and Applications*, Springer Optimization and Its Applications, New York, NY, USA, 2018.

[3] S. Rebennack, A. Arulselvan, L. Elefteriadou, and P. M. Pardalos, "Complexity analysis for maximum flow problems with arc reversals," *Journal of Combinatorial Optimization*, vol. 19, no. 2, pp. 200–216, 2010.

[4] U. Pyakurel, H. N. Nath, S. Dempe, and T. N. Dhamala, "Efficient dynamic flow algorithms for evacuation planning problems with partial lane reversal," *Mathematics*, vol. 7, pp. 1–29, 2019.

[5] P. P. Bhandari and S. R. Khadka, "Evacuation contraflow problems with not necessarily equal transit time on anti-parallel arcs," *American Journal of Applied Mathematics*, vol. 8, pp. 230–235, 2020.

[6] U. Pyakurel and T. N. Dhamala, "Models and algorithms on contraflow evacuation planning network problems," *International Journal of Operations Research*, vol. 12, pp. 36–46, 2015.

[7] U. Pyakurel and T. N. Dhamala, "Continuous time dynamic contraflow models and algorithms," *Advances in Operations Research*, vol. 2016, Article ID 368587, 7 pages, 2016.

[8] S. Kim, S. Shekhar, and M. Min, "Contraflow transportation network reconfiguration for evacuation route planning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 1–15, 2008.

[9] U. Pyakurel, T. N. Dhamala, and S. Dempe, "Efficient continuous contraflow algorithms for evacuation planning problems," *Annals of Operations Research*, vol. 254, no. 1-2, pp. 335–364, 2017.

[10] U. Pyakurel, H. N. Nath, and T. N. Dhamala, "Partial contraflow with path reversals for evacuation planning," *Annals of Operations Research*, vol. 283, no. 1-2, pp. 591–612, 2019.

[11] U. Pyakurel, H. N. Nath, and T. N. Dhamala, "Efficient contraflow algorithms for quickest evacuation planning," *Science China Mathematics*, vol. 61, no. 11, pp. 2079–2100, 2018.

[12] R. C. Dhungana and T. N. Dhamala, "Flow improvement in evacuation planning with budget constrained switching costs," *International Journal of Mathematics and Mathematical Sciences*, vol. 2020, Article ID 1605806, 10 pages, 2020.

[13] U. Pyakurel and S. Dempe, "Network flow with intermediate storage: models and algorithms," *SN Operations Research Forum*, vol. 1, pp. 1–23, 2020.

[14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, NJ, USA, 1993.

[15] M. Skutella, "An introduction to network flows over time," in *Research Trends in Combinatorial Optimization*, pp. 451–482, Springer, Berlin, Germany, 2009.

[16] L. Fleischer and É. Tardos, "Efficient continuous-time dynamic network flow algorithms," *Operations Research Letters*, vol. 23, no. 3-5, pp. 71–80, 1998.

[17] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, USA, 1962.

[18] J. B. Orlin, "A faster strongly polynomial minimum cost flow algorithm," *Operations Research*, vol. 41, no. 2, pp. 338–350, 1993.

[19] R. E. Burkard, K. Dlaska, and B. Klinz, "The quickest flow problem," *ZOR Zeitschrift fur Operations Research Methods and Models of Operations Research*, vol. 37, no. 1, pp. 31–58, 1993.

[20] M. Saho and M. Shigeno, "Cancel-and-tighten algorithm for quickest flow problems," *Networks*, vol. 69, no. 2, pp. 179–188, 2017.

[21] E. Minieka, "Maximal, lexicographic, and dynamic network flows," *Operations Research*, vol. 21, no. 2, pp. 517–527, 1973.

[22] W. L. Wilkinson, "An algorithm for universal maximal dynamic flows in a network," *Operations Research*, vol. 19, no. 7, pp. 1602–1612, 1971.

[23] S. Ruzika, H. Sperber, and M. Steiner, "Earliest arrival flows on series-parallel graphs," *Networks*, vol. 57, no. 2, pp. 169–173, 2011.

[24] H. N. Nath, U. Pyakurel, T. N. Dhamala, and S. Dempe, "Dynamic network flow location models and algorithms for evacuation planning," *Journal of Industrial and Management Optimization*, vol. 13, 2020.

International Journal of Operations Research Vol. 15, No. 4, 175-185 (2018)

Network Flow Approach for Locating Optimal Sink in Evacuation Planning

Hari Nandan Nath[1*], Tanka Nath Dhamala[2]

[1]Tribhuvan University, Bhaktapur Multiple Campus, Bhaktapur, Nepal

[2]Tribhuvan University, Central Department of Mathematics, Kathmandu, Nepal

Abstract: Network flow models have been widely applied for evacuation planning, which involves moving people from risk areas (sources) to safe places (sinks) using some means of transportation, to optimize traffic flow in urban road networks. The decisions related to the locations of the sinks are also important to maximize the number of evacuees or minimize time for the evacuees to reach the safe places. In this work, we consider the problems of identifying the optimal sink node out of a given set of possible sink-nodes in a single source network to maximize the flow value, and that to minimize the time to transfer a given flow value to the sink in minimum time. Designing efficient computational procedures to solve the problems, we prove that the problems can be solved with strongly polynomial time complexity. Corresponding optimal sink location problems along with identification of ideal direction of the flow based on contraflow approach are also solved in strongly polynomial time. Our results are substantiated by a case illustration based on Kathmandu road network.

Keyword — Evacuation planning, sink location, network flow, dynamic flow, maximum flow, quickest flow, contraflow

1. INTRODUCTION

Evacuation planning decisions involve the decisions related to the traffic flow on urban road networks to transfer people from danger areas to safe places or shelters. Choosing optimal shelters out of a given number of shelters is an important optimization problem, and is a growing research area. Likewise, reversing the direction of usual traffic flow so as to increase the flow towards the shelters is also one of the active research problems in evacuation planning. We discuss, briefly, some of the works done in these directions.

Sherali, Carter, and Hobeika (1991) develop a location-allocation model to choose a set of shelter locations from among feasible locations. Their approach is a discrete minisum or median location approach, in which they minimize the total time spent by evacuees. They use the congestion related travel time developed by U.S. Bureau of Public Roads (BPR) (1964). Noting that the problem is NP-hard, they develop a fast heuristic which performs well in practice and also present exact enumeration algorithms with running time increase rapidly with the growing size of the problem.

Kongsomsaksakul, Chen, and Yang (2005) propose a bilevel model in which the upper level (representing the planner) chooses the number of shelters and their locations from among a given set of potential locations with the objective to minimize the total evacuation time. The lower level (representing evacuees) is a combined distribution and assignment (CDA) which is formulated with a principle that evacuees try to travel to safe shelter with the least travel time. They also use the travel time developed by BPR. Realizing the difficulty of solving the bilevel programming problems analytically, they develop a genetic algorithm (GA) to solve the problem. A similar approach can be found in Ng, Park, and Waller (2010) in which the lower level problem is a deterministic user equilibrium (DUE) (Sheffi (1985)) in which the evacuees find their shortest routes to the shelters they are assigned to.

For the evacuees who depend on transit vehicles, Goerigk, Grün, and Heßler (2014) develop an integer programming model to find locations for optimal shelters along with the schedule of buses from pick-up locations to shelters. Realizing the NP-completeness of the problem, they develop a branch-and-price approach to solve the problem. Considering the movement of transit-dependent evacuees and evacuees having individual vehicles, Goerigk, Deghdak, and Heßler (2014) present a comprehensive evacuation plan. They formulate the problem as a multicommodity flow, multicriteria mixed-integer programming problem with the objectives to minimize the number of used shelters, evacuation time, and the total risk exposure of evacuees. Given a risk value to each arc, the total risk value is the sum the product

*Corresponding author's e-mail: hari672@gmail.com

http://doi.org/10.6886/IJOR.201812_15(4).0003

of number of people moving on the arc and risk value of the arc. They propose a heuristic solution procedure based on Genetic Algorithms (GA) because the mixed-integer programming problem is not likely to be solvable in sufficient time for real-world instances.

Based on network flow approach, Heßler and Hamacher (2016) present a mixed integer programming model in which given a supply at each node of a network, and cost of opening a shelter at a node, they minimize the total cost of the opened shelter. They present exact algorithms for choosing single/multiple sink(s), in cases when flows originating at different nodes do not add up on edges, and sinks are uncapacitated, approximation algorithms for the cases when flows originating at different nodes add up on edges and on the chosen sinks as well.

In case of emergency evacuation, since the traffic flow is directed towards the safe areas, the road segments on the direction of paths towards the danger areas (sources) are little occupied or empty while the segments in the opposite direction are over-occupied. Contraflow problems focus on addressing such situations, mathematically, to optimize traffic flow with the ideal direction of arcs in the transportation network.

To solve a multi-source, multi-sink contraflow configuration problem, Kim, Shekhar, and Min (2008) present algorithms based on greedy heuristic and bottleneck relief heuristic. Wang, Wang, Zhang, Ip, and Furuta (2013) considered a multi-objective optimization model to minimize the evacuation time of different categories of evacuees from a single source to appropriate (multiple) shelters and to minimize the traffic set-up time to reverse traffic flow. Their algorithm uses discrete version of particle swarm optimization (PSO) metaheuristic. Zhao, Feng, Li, and Bernard (2016) propose a bi-level model in which a tabu search algorithm is applied to find an optimal lane reversal plan in the upper-level, and the lower-level utilizes a simulated annealing algorithm for lane-based route plans with intersection crossing conflict elimination.

Apart from heuristic techniques, there are available exact analytical algorithms also. Rebennack, Arulselvan, Elefteriadou, and Pardalos (2010) introduce efficient algorithms to solve the maximum dynamic contraflow and the quickest contraflow problems in a single source single sink network. T. Dhamala and Pyakurel (2013) present strongly polynomial time algorithm for the earliest arrival (also known as universally maximum) contraflow problem on a series parallel network. Algorithms to solve the lexicographically maximum contraflow problem on multi-source-multi-sink network and the earliest arrival contraflow problem on multi-source-single-sink network are presented in Pyakurel and Dhamala (2015). Pyakurel and Dhamala (2017b) present a pseudo-polynomial time algorithm to solve the problem on two terminal general network. These problems are solved for discrete time set up. The discrete time algorithms are converted to continuous time algorithms by Pyakurel and Dhamala (2016, 2017a) using the idea of natural transformations in Fleischer and Tardos (1998). Algorithms for contraflow approach for quickest flow with constant and load dependent time on arcs can be found in Pyakurel, Nath, and Dhamala (2018a). Realizing the need of saving capacities of arcs for other facilities during evacuation, Pyakurel, Nath, and Dhamala (2018b) apply partial contraflow technique in a network with path reversal capabilities.

The paper is organized as follows. In Section 2, we present the necessary network flow models on which our approach is based. We introduce optimal sink models and solution procedures in Section 3. Section 4 deals with the identification of optimal sinks allowing arc reversals, and Section 6 concludes the paper.

2. BASIC IDEAS

In this section, we present basic mathematical tools used for developing models and solution algorithms in this work. For modeling, we consider road segments as arcs and their intersections as nodes of a directed network. Anything that moves on arcs is referred to as flow. The amount of flow that can enter a road segment per unit time is the capacity of the arc and the time the flow takes to travel from one end to the other end of the road segment represents the travel time on the corresponding arc.

2.1 Static and dynamic flows

Let $N = (V, A)$ be a directed network where V is the set of nodes and A is the set of arcs with $|V| = n$ and $|A| = m$. For a node $i \in V$, we denote the set of outgoing arcs from i by $A_i^{\text{out}} = \{e \in A : e = (i, j) \text{ for some } j \in V\}$, and the set of incoming arcs to i by $A_i^{\text{in}} = \{e \in A : e = (j, i) \text{ for some } j \in V\}$. For each $e \in A$, b_e, τ_e denote the upper capacity (or capacity), and travel time on e. A flow travels from a special $S \subset V$, called source nodes to another $D \subset V$, called sink nodes. We represent such a network by $N = (V, A, b, \tau, S, D)$. If $S = \{s\}, D = \{d\}$, we write $N = (V, A, b, \tau, s, d)$.

A static s-d flow $x : A \to \mathbb{R}_{\geq 0}$ is a function of non-negative values such that $x(e) = x_e$ satisfying the following.

$$\sum_{e \in A_i^{\text{out}}} x_e - \sum_{e \in A_i^{\text{in}}} x_e = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = d \\ 0 & \text{if } i \in V \setminus \{s, d\} \end{cases} \tag{1}$$

$$0 \leq x_e \leq b_e \tag{2}$$

The quantity v is the value of the static flow x which is the flow that goes out of the source s and reaches the sink d. The third set of constraints in (1) states that whatever flow comes to a node other than the source or the sink goes out of it. The constraints set (2) restricts the flow in any arc exceed its capacity. The maximum static flow problem seeks to maximize v under the constraints (1) - (2) for the solution of which there are available highly efficient algorithms because of the special structure of the linear program stated above. For more details, see T. N. Dhamala, Pyakurel, and Dempe (2018).

A dynamic flow x^{dyn} with time horizon T consists of a Lebesgue- integrable functions $x_e^{\text{dyn}} : [0, T) \to \mathbb{R}_{\geq 0}$ for each arc $e \in A$ with $x_e^{\text{dyn}}(t) = 0$ for $t \geq T - \tau(e)$ and satisfies the following.

$$\sum_{e \in A_i^{\text{in}}} \int_0^{t-\tau(e)} x_e^{\text{dyn}}(\theta) d\theta - \sum_{e \in A_i^{\text{out}}} \int_0^{t} x_e^{\text{dyn}}(\theta) d\theta \geq 0, \ \forall t \in [0, T), \forall i \in V \setminus \{s, d\} \tag{3}$$

$$\sum_{e \in A_i^{\text{in}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta) d\theta - \sum_{e \in A_i^{\text{out}}} \int_0^{T} x_e^{\text{dyn}}(\theta) d\theta = 0, \ \forall i \in V \setminus \{s, d\} \tag{4}$$

$$0 \leq x_e^{\text{dyn}}(t) \leq b_e, \ \forall e \in A, \forall t \in [0, T). \tag{5}$$

$x_e^{\text{dyn}}(\theta)$ is the rate of flow that enters arc e at time θ. The value of the dynamic flow x^{dyn} for the time horizon T is:

$$\begin{aligned}\text{val}_T(x^{\text{dyn}}) &= \sum_{e \in A_d^{\text{in}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta) d\theta - \sum_{e \in A_d^{\text{out}}} \int_0^{T} x_e^{\text{dyn}}(\theta) d\theta \\ &= \sum_{e \in A_s^{\text{out}}} \int_0^{T-\tau(e)} x_e^{\text{dyn}}(\theta) d\theta - \sum_{e \in A_s^{\text{in}}} \int_0^{T} x_e^{\text{dyn}}(\theta) d\theta \end{aligned} \tag{6}$$

For a given T, the maximum dynamic flow problem seeks to maximize $\text{val}_T(x^{\text{dyn}})$ under the constraints (3) - (5). The constraints (3) state that there can be a holdover of the flow in the intermediate nodes which ultimately gets cleared by the time horizon T as stated in (4). The constraints in (5) do not allow the flow value to exceed the upper capacity at any time. For more details, we refer to Skutella (2009). If we replace each of the integral by the corresponding summation over the discretized set of times $\{0, \ldots, T\}$, we get the discrete-time version of the problem.

3. IDENTIFICATION OF THE OPTIMAL SINK

In a network with a single source, given a set of feasible sink nodes, we take a straightforward approach for the choice of a single sink depending on the objective of the evacuation problem. If the objective is to send as much flow as possible, we choose the sink which maximizes flow value. Likewise, when a given flow value is given, we choose the sink so as to minimize the time for the flow to reach the sink.

3.1 Optimal sink to maximize the flow value

Definition 1 (MaxStatic sink, MaxDynamic sink). Let $N = (V, A, b, \tau, s)$ be a network with a set of feasible sinks $D \subset V \setminus \{s\}$ and let $v_{stat}(d), v_{dyna}^T(d)$ denote the value of maximum static flow, and maximum dynamic flow with time horizon T, respectively, from s to $d \in D$. We call the node $d_{stat} = \arg\max_{d \in D}\{v_{stat}(d)\}$, the MaxStatic sink and $d_{dyna}^T = \arg\max_{d \in D}\{v_{dyna}^T(d)\}$, the MaxDynamic sink.

Example 1. We consider an evacuation network in Figure 1. Let the source node be s and the set of feasible sinks $D = \{d_1, d_2, d_3\}$. If node d_1 is taken as the sink, the maximum static flow value is 6 (4 via path $s - d_1$, 1 each via $s - d_2 - d_1$, and $s - d_2 - d_3 - d_1$). The dynamic flow value with 2 as the sink and discrete time horizon $T = 3$ is 2 (1 via $s - d_2 - d_1$ twice). The values corresponding to other sinks and time horizons are listed in the following table. So, one can easily conclude that $d_{stat} = 4$ and $d_{dyna}^3 = 3, d_{dyna}^8 = 2, d_{dyna}^{12} = 4$.

d	$v_{stat}(d)$	$v_{dyna}^3(d)$	$v_{dyna}^8(d)$	$v_{dyna}^{12}(d)$
d_1	6	2	30	54
d_2	4	9	28	44
d_3	7	1	28	56

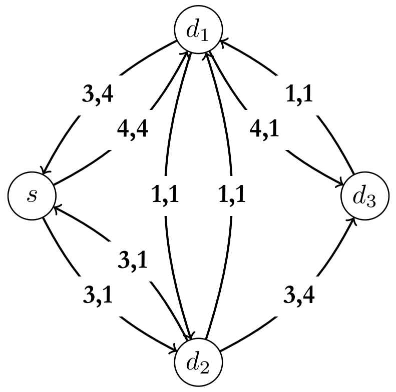


Figure 1: Evacuation Network with arc labels capacity, travel time

Now, we present mathematical programming formulation to identify the MaxStatic sink. We consider a network $N = (V, A, b, \tau, s)$ with a set of feasible sinks D. For the modeling purpose, we consider that there is only one arc incoming to each $d \in D$. This does not restrict the general case because we can always add a node d' to V, and (d, d') to A such that $\tau(d, d') = 0, b(d, d') = \infty$ and replace $d \in D$ by d' so that d is identified with d'. Practically, infinite capacity of (d, d') can be replaced by $\sum_{e\in A_d^{\text{in}}} b_e$. Figure 2 shows such a network transformation in which D becomes $\{d_1', d_2', d_3'\}$.

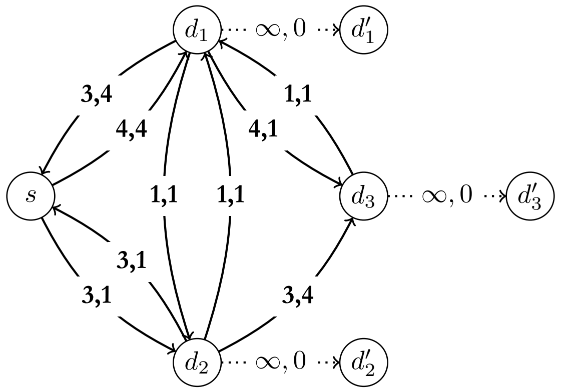


Figure 2: Transformation of the network in Figure 1

Let $A' = \{(i, d) \in A : d \in D\}$. We present the problem of finding MaxStatic sink as a mixed binary integer formulation as:

$$\max \quad \sum_{e\in A'} x_e \tag{7a}$$

$$\sum_{e\in A_i^{\text{out}}} x_e - \sum_{e\in A_i^{\text{in}}} x_e = 0, \ \forall i \in V \setminus (\{s\} \cup D) \tag{7b}$$

$$x_e \leq b_e, \ \forall e \in A \setminus A' \tag{7c}$$

$$x_e \leq b_e y_e, \ \forall e \in A' \tag{7d}$$

$$\sum_{e\in A'} y_e = 1 \tag{7e}$$

$$y_e \in \{0, 1\} \tag{7f}$$

The objective (7a) maximizes the flow entering to the sinks. Because of (7d)-(7f), the flow will be directed towards only one sink. Constraints (7b) are mass balance constraints in the intermediate nodes. (7c),(7d) are capacity constraints. The constraint (7e) chooses only one sink out of feasible sinks.

In the constraint set (7d), since b_e may not be in $\{-1, 0, +1\}$, the matrix associated with the constraints is not totally unimodular. Thus the linear programming relaxation of the above integer programming may not give the integral solution. However, if $y_e, e \in A'$ is bound to be binary, we have the following observation.

Theorem 1. If $b_e \in \mathbb{Z}$, the mixed integer programming (7a)-(7f) has all integral solutions.

Proof. Let $I^{|D|}$ denote the set of the column vectors of the $|D| \times |D|$ identity matrix. Because of our assumption, indegree of each $d \in D$ is 1, and hence the solution set of the constraints (7e) and (7f) is $I^{|D|}$. If $y = [y_e : e \in A']$, then the column vector y has exactly $|D|$ components. If a fix $y \in I^{|D|}$ is chosen, the mixed integer programming problem (7) becomes a linear programming problem. Since $|I^{|D|}| = |D|$, MIP (7) can be solved by solving linear program (7a-7d) at most $|D|$ times.

Further, let the matrix equation $Mx = 0$ represent the constraints (7b). If e has both of its ends in $V \setminus (\{s\} \cup D)$, then the column of M corresponding to x_e will have exactly two entries $+1, -1$. Each of the other columns of M corresponding to x_e with an end of e in $\{s\} \cup D$ will have exactly one entry either 1 or -1. This shows that M is totally unimodular. Since the polyhedron $\{x \in \mathbb{R}^n : Mx = b, 0 \leq x \leq u\}$ with totally unimodular M and integer u is an integral polyhedron, we can get all integer solutions of the mixed integer programming (7) if $b_e \in \mathbb{Z}, \forall e \in A$. □

The above is particularly important, because most of the network flow algorithms use integrality of flow to develop efficient algorithms. We present a straight-forward procedure to identify MaxStatic sink in Algorithm 1 which iteratively chooses an element $d \in D$, finds the maximum static s-d flow value, and selects d as the MaxStaic sink if the flow-value is improved. When the iteration ends, the algorithm returns the MaxStatic sink and the corresponding static flow.

Algorithm 1: Locating the MaxStatic sink

```
Input  : Directed network $N = (V, A, b, s)$, the set of possible sink locations $D$
Output: Optimal sink $d^*$, the corresponding static flow $x$
1  $curr\_max\_v = -1$
2  for $d \in D$ do
3      $new\_max\_v = v_{stat}(d)$
4      if $new\_max\_v > curr\_max\_v$ then
5          $d^* = d$
6          $curr\_max\_v = new\_max\_v$
7          $x =$ corresponding static flow
8      end
9  end
10 return $d^*, x$
```

The practical running time of Algorithm 1 can be improved by finding $v_{stat}(d)$ in Line 3 only if $\sum_{e \in A_d^{\text{in}}} b_e > curr\_max\_v$ in the network without the transformation mentioned in Figure 2, and exiting the **for** loop and returning d as d^* if $v_{stat}(d) = \sum_{e \in A_s^{\text{out}}} b_e$.

Algorithm 1 leads to the following important implication.

Theorem 2. The problem of identifying MaxStatic sink can be solved with strongly polynomial time complexity of $O(nm|D|)$.

Proof. Let C be the complexity of a maximum static flow calculation. Since Algorithm 1 terminates after $|D|$ iterations and Line 3 calculates a maximum static flow value in each iteration, the complexity of the algorithm is $O(C|D|)$. Using the algorithm of Orlin (2013), the maximum static flow calculation can be done in $O(nm)$ time, where n, m denote the number of nodes and number of arcs in N. Hence, Algorithm 1 solves the problem in $O(nm|D|)$ time. This proves the assertion. □

Given a time horizon T, Ford and Fulkerson (1962) showed that for a dynamic maximum flow problem, the maximum flow can be obtained by temporally repeating the static flow. The following result, for the continuous time case, connects the dynamic flow with its static counterpart. For the discrete-time version, time T is replaced by $T + 1$.

Lemma 1 (Fleischer and Tardos (1998); Skutella (2009))**.** Let x be a feasible static s-d flow with value v, then the value of the corresponding temporally repeated dynamic flow is equal to $Tv - \sum_{e \in A} \tau_e x_e$.

Using the result in Lemma 1, we can replace the objective (7a) by $\sum_{e \in A'} Tx_e - \sum_{e \in A} \tau_e x_e$ to obtain the MaxDynamic sink, in continuous time setting. In the discrete time setting, we can just replace T by $T + 1$. The result analogous to that in Theorem 1 is valid in this case also.

Replacing $v_{stat}(d)$ by $v_{dyna}^T(d)$ in Line 3, we can adapt Algorithm 1 to identify the MaxDynamic sink.

Theorem 3. The problem of identifying MaxDynamic sink can be solved with strongly polynomial time complexity of $O((m \log n)(m + n \log n)|D|)$.

Proof. For a given sink $d \in D$, let N' be the network obtained by adding an arc (d, s) to N with $b(d, s) = \infty, \tau(d, s) = -T$ in a continuous-time setting (in the discrete time setting, we take $\tau(d, s) = -(T + 1)$). Assuming that there is no directed path from d to s in N, let x be the the min-cost circulation in N', then as a result of Lemma 1, the restriction of x to N is the temporally repeated static flow, with value v, corresponding to the maximum dynamic flow and $v^T_{dyna}(d) = Tv - \sum_{e\in A} \tau_e x_e$ (in discrete-time setting, $v^T_{dyna}(d) = (T + 1)v - \sum_{e\in A} \tau_e x_e$). As minimum cost flow problem, using enhanced capacity scaling technique, can be solved in $O((m \log n)(m+n \log n))$ time (Orlin (1993)), and the MaxDynamic sink can be identified iterating over elements of D, we get the required result. □

3.2 Optimal sink to minimize the quickest time

Given supply F at s, the quickest flow problem seeks to find the dynamic s-d flow which has the value F within the minimum time horizon. The following result is crucial in this regard.

Theorem 4 (Lin and Jaillet (2015))**.** The quickest flow problem in $N = (V, A, b, \tau, s, d)$ with a given supply at s can be formulated as the following linear programming problem

$$\min \quad \frac{F + \sum_{e\in A} \tau_e \cdot x_e}{v} \tag{8a}$$

$$\sum_{e\in A_i^{\text{out}}} x_e - \sum_{e\in A_i^{\text{in}}} x_e = \begin{cases} v, & \text{if } i = s \\ -v, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \tag{8b}$$

$$0 \le x_e \le b_e \; \forall e \in A \tag{8c}$$

Proof. Let x be a feasible static s-d flow with value v. Then the value of the corresponding dynamic flow with time horizon T is $Tv - \sum_{e\in A} \tau_e x_e$ (see Lemma 1). So, we can formulate the maximum dynamic flow problem with a time horizon T as:

$$\max \quad Tv - \sum_{e\in A} \tau_e x_e \tag{9a}$$

$$\sum_{e\in A_i^{\text{out}}} x_e - \sum_{e\in A_i^{\text{in}}} x_e = \begin{cases} v, & \text{if } i = s \\ -v, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \tag{9b}$$

$$0 \le x_e \le b_e \; \forall e \in A \tag{9c}$$

Let $\psi(T) = \max Tv - \sum_{e\in A} \tau_e x_e$ under constraints (9b)-(9c). Then $\psi(T)$ is non-decreasing function of T (Burkard, Dlaska, and Klinz (1993)). So, the quickest flow problem with a given supply F is equivalent to finding the minimum time T^* such that $\psi(T^*) \geq F$. Hence, the quickest flow problem can be formulated as:

$$\min \quad T \tag{10a}$$

$$Tv - \sum_{e\in A} \tau_e x_e \geq F \tag{10b}$$

along with (9b)-(9c). The constraint (10b) can be reshuffled as:

$$T \geq \frac{F + \sum_{e\in A} \tau_e x_e}{v} \tag{11}$$

in which $v \neq 0$ because if $v = 0$, no flow can be sent from the source to the sink. Since (9b)-(9c) do not involve T, we can replace T in (10a) with $\frac{F+\sum_{e\in A} \tau_e x_e}{v}$ and the result follows. □

Given a source s and a supply F, if a sink is to be chosen from a given set of feasible sets, then we take a point of view to choose the sink which minimizes the quickest time.

Definition 2 (Quickest sink)**.** Let $N = (V, A, b, \tau, s)$ be a network with a set of feasible sinks $D \subset V$ and let $T^F(d)$ denote the quickest time to transport the flow value of F from s to $d \in D$. We call the node $\arg\min_{d\in D}\{T^F(d)\}$, the Quickest sink.

We can adapt the mathematical formulation given in (7), replacing its objective by (8a) where $v = \sum_{e\in A'} x_e$. To linearize the objective (8a), we can introduce a new variable $x'_e = x_e/v$ and adjust the constraints accordingly converting the problem into mixed binary integer linear program.

We present a simple procedure to identify the quickest sink in Algorithm 2. The algorithm iteratively chooses an element $d \in D$, finds the quickest time to send F from s to d, and selects d as the Quickest sink if the quickest time decreases. When the iteration ends, the algorithm returns the Quickest sink and the static flow corresponding to the temporally repeated quickest flow.

Algorithm 2: Locating the Quickest sink

```
   Input  : directed network $N = (V, A, b, s, \tau)$, supply $F$ at $s$, the set of possible sink locations $D$
   Output: optimal sink $d^*$, the corresponding static flow $x$
1  $curr\_quickest\_time = \infty$
2  for $d \in D$ do
3      $new\_quickest\_time = T^F(d)$
4      if $new\_quickest\_time < curr\_quickst\_time$ then
5          $d^* = d$
6          $curr\_quickest\_time = new\_quickest\_time$
7          $x$ = corresponding static flow
8      end
9  end
10 return $d^*, x$
```

On the basis of Algorithm 2, we have the following result.

Theorem 5. The Quickest sink can be computed in strongly polynomial time complexity of $O(nm^2 \log^2 n)|D|$.

Proof. There are $|D|$ iterations in Algorithm 2 in each of which it calculates the quickest flow. Saho and Shigeno (2017) adapted cancel-and-tighten algorithm of finding minimum cost flow to compute quickest flow, which has the complexity of $O(nm^2 \log^2 n)$. Hence, Quickest sink can be identified in $O(nm^2 \log^2 n|D|)$ time. □

4. OPTIMAL SINK WITH ARC REVERSAL STRATEGY

4.1 Contraflow approach

The contraflow approach reverses the direction of necessary arcs to optimize the flow in a directed network. The common procedure to do so is to solve the corresponding problem in, what is known as, an auxiliary network. The auxiliary network of $N = (V, A, b, \tau, s, d)$ is $\bar{N} = (V, \bar{A}, \bar{b}, \bar{\tau}, s, d)$ where

$$\bar{A} = \{(i, j) : (i, j) \in A \text{ or } (j, i) \in A\}$$

and for each $(i, j) \in \bar{A}$,

$$\begin{aligned} \bar{b}(i,j) &= b(i,j) + b(j,i) \\ \bar{\tau}(i,j) &= \begin{cases} \tau(i,j) & \text{if } (i,j) \in A \\ \tau(j,i) & \text{otherwise} \end{cases} \end{aligned}$$

in which we consider $b(i, j) = 0$ whenever $(i, j) \notin A$, and vice versa. Figure 3 is the auxiliary network constructed for the network in Figure 1.

4.2 Optimal sink with contraflow

Because of the change in capacity of the arcs, the decisions related to optimal sink also change if we apply contraflow approach. Example 2 illustrates this fact.

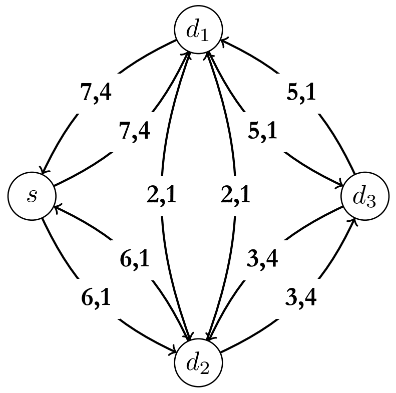


Figure 3: Auxiliary network of the network in Figure 1

Example 2. Consider the network considered in Example 1. To allow arc reversal, we construct the auxiliary network as depicted in Figure 3. Taking s as source, if $d = d_1$, the maximum static flow value is 12 (7 via $s - d_1$, 3 via $s - d_2 - d_3 - d_1$, 2 via $s - d_2 - d_1$). The corresponding values with $d = 3, 4$ are 11 and 8, respectively. Thus MaxStatic sink allowing lane reversals is the node 2 while it is node 4 without allowing arc reversals with maximum static flow value 7 (see Example 1). Considering 2 as the sink, the maximum static flow x allowing arc reversals is given in the following table.

e	(s, d_1)	(s, d_2)	(d_1, s)	(d_1, d_3)	(d_1, d_2)	(d_2, s)	(d_2, d_1)	(d_2, d_3)	(d_3, d_1)	(d_3, d_2)
$x(e)$	7	5	0	0	0	0	2	3	3	0

Since $x(s, d_1) > b(s, d_1), x(s, d_2) > b(s, d_2), x(d_2, d_1) > b(d_2, d_1), x(d_3, d_1) > b(d_3, d_1)$, the arcs to be reversed are: $(d_1, s), (d_2, s), (d_1, d_2)$ and (d_1, d_3).

Given a network $N = (V, A, b, \tau, s)$ and set of feasible sinks D, to identify the MaxStatic sink, MaxDynamic sink, Quickest sink allowing arc reversals, we solve the corresponding problem in $\bar{N}$. We present Algorithm 3 in which the input is $N = (V, A, b, \tau, s)$ and a set of feasible sinks D. For the calculation of the quickest sink, the supply F at the source s is also be given. In Line 1, the the auxiliary network $\bar{N} = (V, \bar{A}, \bar{b}, \bar{\tau}, s)$ is constructed. In Line 2, depending on the objective, MaxStatic sink, MaxDynamic sink, or Quickest sink is identified as described in Section 3. The corresponding static flow x is also calculated. In Line 3, x is decomposed into paths and cycles and cycle flows are removed so that the set of arcs to be reversed in Line 4 is well-defined. After the removal of cycle flows, if an arc in A has flow value more than its original capacity, its opposite arc will be reversed. If the solution shows any positive flow in any arc not in A, then the corresponding opposite arc is also reversed. In this way, Line 4 gives the set of arcs to be reversed.

Algorithm 3: Locating sink with contraflow

```
  Input  : directed network $N = (V, A, b, \tau, s)$, the set of possible sink locations $D$ (and supply $F$ at the
           source $s$ in case of Quickest sink) with arc reversal capability
  Output: optimal sink $d^*$, set of arcs to be reversed, corresponding static flow $x$
1 Construct the auxiliary network $\bar{N}$.
2 Solve the corresponding problem in $\bar{N}$ to find the optimal sink $d^*$, and the corresponding static flow $x$.
3 Decompose $x$ into paths and cycles and remove cycles.
4 $R = \{(j, i) \in A : x(i, j) > b(i, j)$ if $(i, j) \in A$ or $x(i, j) > 0$ if $(i, j) \notin A\}$
5 return $d^*, R, x$.
```

Theorem 6. The problems of identification of MaxStatic sink, MaxDynamic sink, and Quickest sink allowing arc reversals can be solved in strongly polynomial time with the complexity of the corresponding problems without allowing lane reversals.

Proof. In Algorithm 3, the auxiliary network, in Line 1, can be constructed in $O(m)$ times. Line 2 finds MaxStatic sink, MaxDynammic sink, or Quickest sink in the auxiliary network depending on the problem. So, the complexity of Line 2 is $O(C|D|)$ where C is

(i) $O(mn)$, the complexity of the maximum static flow calculation, in case of the MaxStatic sink

(ii) $O((m \log n)(m + n \log n))$, the complexity of the minimum cost flow calculation, in case of the MaxDynamic sink

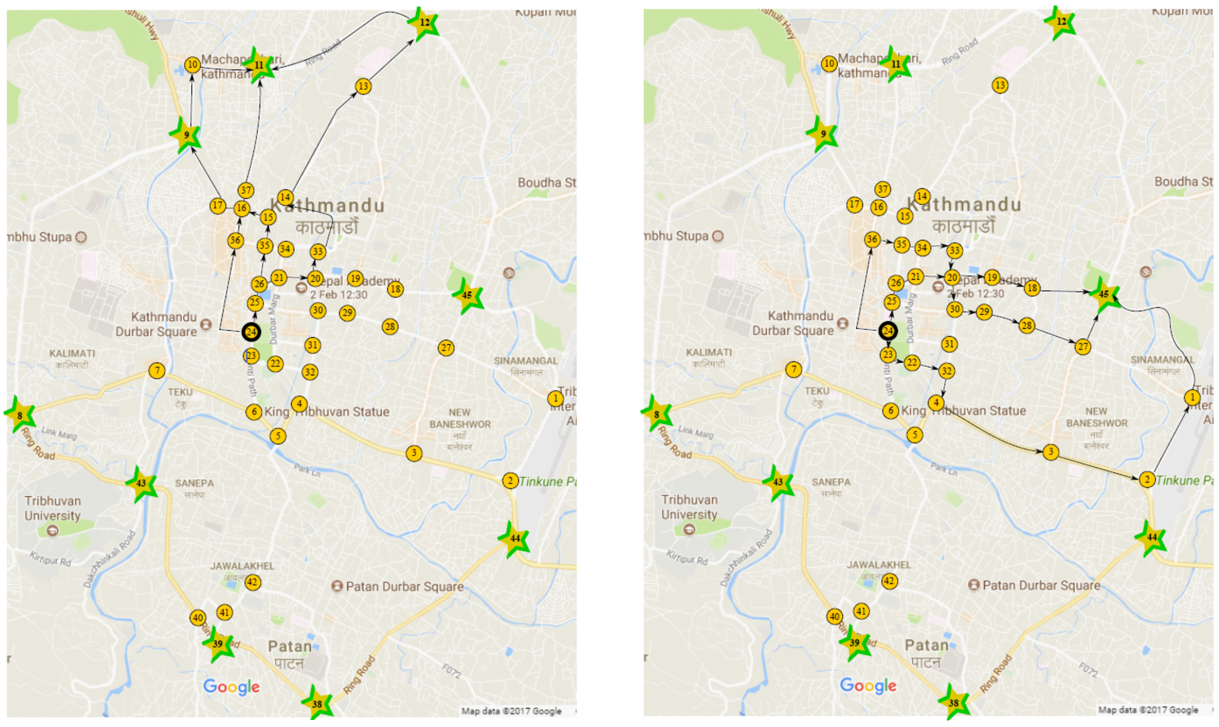
(iii) $O(nm^2 \log^2 n)$, the complexity of quickest flow calculation, in case of the Quickest sink

The decomposition of a static flow into paths and cycles can be done in $O(nm)$ time (see Ahuja, Magnanti, and Orlin (1993)). So the time complexity of Line 3 in Algorithm 3 is $O(mn)$. The construction of R in Line 4 requires $O(m)$ comparisons. Hence, the overall complexity of the algorithm is dominated by the complexity of Line 2, which is $O(C|D|)$. This proves the assertion.

□

5. CASE ILLUSTRATION

As an illustration, we consider Kathmandu road network within Ring Road only with major road segments (cf. Figure 4). We take the node adjoining Tundikhel area (denoted in the figure by 24) as the source and Kalanki (8), Balaju (9), Gongabu (11), Narayan Gopal Chowk (12), Gaushala (45), Koteshwar Jadibuti (44), Satdobato (38), Ekantakuna (39), and Balkhu (43) as possible sinks. To implement auto-based evacuation planning, we take the capacities of arcs between 2 cars per second to 4 cars per second depending on the width of the segment. The direction of the usual traffic flow is taken as the direction of the arc. The time related to an arc is taken using Google Maps.



(a) (b)

Figure 4: Kathmandu network with (a) Node 11 as MaxDynamic sink (b) Node 45 as MaxDynamic sink with contraflow

Considering a time horizon $T = 1$ hour, we find the Maxdynamic sink as Gongabu(11) with the dynamic flow value of 12,120 cars. However, if we allow arc reversal, the Maxdynamic sink is Gaushala(45) with the corresponding flow value 27,360 cars. If we take a time horizon of $T = 2$ hours, the sink locations with and without arc reversal are respectively the same as those of $T = 1$ hour with flow values 33,720 and 70,560.

The Quickest sink locations with different values of F are listed in the following table.

F	Without contraflow		With contraflow	
	Quickest sink	Quickest time (sec)	Quickest sink	Quickest time (sec)
1,000	Balaju (9)	1,160	Balaju (9)	910
10,000	Gongabu (11)	3,247	Gaushala(45)	2,150
20,000	Gongabu (11)	4,913	Gaushala(45)	2,987

6. CONCLUSION

In this work, the problem of identification of an optimal sink from among a given set of uncapacitated sinks is considered with and without allowing arc reversals, with different aspects of network flow, viz. maximum static flow, maximum dynamic flow, and quickest flow. Presenting the solution procedures, it is proved that the problems can be solved in strongly polynomial time. The problem considered is particularly important in case of evacuation planning when a single shelter has to be chosen from among given shelters of sufficient capacities. One can use MaxStatic sink, when a maximum number of evacuees are to be sent as one wave, MaxDynamic sink, when the maximum number of evacuees are to be sent within a given time horizon, and Quickest sink when a given number of evacuees are to be evacuated as quickly as possible.

ACKNOWLEDGMENTS

The authors acknowledge partial supports of Alexander von Humboldt Foundation (AvH) and German Academic Exchange Service (DAAD). The first author acknowledges the support of University Grants Commission (UGC) Nepal also for granting PhD fellowship and research support.

REFERENCES

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications.* Prentice Hall.

Bureau of Public Roads. (1964). *Traffic assignment manual.* Washington D.C.: US Department of Commerce, Urban Planning Division.

Burkard, R. E., Dlaska, K., & Klinz, B. (1993). The quickest flow problem. *ZOR - Meth. & Mod. of OR*, *37*(1), 31–58. Retrieved from `https://doi.org/10.1007/BF01415527`

Dhamala, T., & Pyakurel, U. (2013). Earliest arrival contraflow problem on series-parallel graphs. *International Journal of Operations Research*, *10*(1), 1–13.

Dhamala, T. N., Pyakurel, U., & Dempe, S. (2018). A critical survey on the network optimization algorithms for evacuation planning problems. *International Journal of Operations Research*, *15*(3), 101–133.

Fleischer, L., & Tardos, E. (1998). Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, *23*(3), 71–80.

Ford, F., & Fulkerson, D. (1962). *Flows in networks.* New Jersey: Princeton University Press.

Goerigk, M., Deghdak, K., & Heßler, P. (2014). A comprehensive evacuation planning model and genetic soution algorithm. *Transportation Research, Part E*, *71*, 82–97.

Goerigk, M., Grün, B., & Heßler, P. (2014). Combining bus evacuation with location decisions: A branch-and-price approach. *Transportation Research Procedia*, *2*, 783–791.

Heßler, P., & Hamacher, H. (2016). Sink location to find optimal shelters in evacuation planning. *EURO Journal on Computational Optimization*, *4*(3-4), 325–347.

Kim, S., Shekhar, S., & Min, M. (2008). Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, *20*(8), 1115–1129.

Kongsomsaksakul, S., Chen, A., & Yang, C. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, *6*, 4237–4252.

Lin, M., & Jaillet, P. (2015). On the quickest flow problem in dynamic networks: a parametric mincost flow approach. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms* (pp. 1343–1356).

Ng, M., Park, J., & Waller, S. (2010). A hybrid bilevel model for the optimal shelter assignment in emergency evacuations. *Computer-Aided Civil and Infrastructure Engineering*, *25*(8), 547–556.

Orlin, J. (1993). A faster strongly polynomial minimum cost flow algorithm. *Operations research*, *41*(2), 338–350.

Orlin, J. (2013). Max flows in $o(nm)$ time, or better. , 765–774.

Pyakurel, U., & Dhamala, T. (2015). Models and algorithms on contraflow evacuation planning network problems. *International Journal of Operations Research*, *12*(2), 36–46.

Pyakurel, U., & Dhamala, T. (2016). Continuous time dynamic contraflow models and algorithms. *Advances in Operations Research.*

Pyakurel, U., & Dhamala, T. (2017a). Continuous dynamic contraflow approach for evacuation planning. *Annals of Operations Research*, *253*(1), 573–598.

Pyakurel, U., & Dhamala, T. (2017b). Evacuation planning by earliest arrival contraflow. *Journal of Industrial & Management Optimization*, *13*(1), 489–503.

Pyakurel, U., Nath, H., & Dhamala, T. (2018a). Efficient contraflow algorithms for quickest evacuation planning. *Science China Mathematics*, *61*(11), 2079–2100.

Pyakurel, U., Nath, H., & Dhamala, T. (2018b). Partial contraflow with path reversals for evacuation planning. *Annals of Operations Research.*

Rebennack, S., Arulselvan, A., Elefteriadou, L., & Pardalos, P. (2010). Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, *19*(2), 200–216.

Saho, M., & Shigeno, M. (2017). Cancel-and-tighten algorithm for quickest flow problems. *Networks*, *69*(2), 179–188.

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Englewood Cliffs, NJ: Prentice-Hall.

Sherali, H., Carter, T., & Hobeika, A. (1991). A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B: Methodological*, *25*(6), 439–452.

Skutella, M. (2009). An introduction to network flows over time. In W. Cook, L. Lovász, & J. Vygen (Eds.), *Research trends in combinatorial optimization* (pp. 451–482). Springer.

Wang, J., Wang, H., Zhang, W., Ip, W., & Furuta, K. (2013). Evacuation planning based on the contraflow technique with consideration of evacuation priorities and traffic setup time. *IEEE Transactions on Intelligent Transportation Systems*, *14*(1), 480–485.

Zhao, X., Feng, Z., Li, Y., & Bernard, A. (2016). Evacuation network optimization model with lane-based reversal and routing. *Mathematical Problems in Engineering*.

Optimization Models and Algorithms for Evacuation Planning: The Bus Evacuation Problem

Hari Nandan Nath
hari_672@gmail.com
Tribhuvan University,Kathmandu, Nepal

Introduction

Every human activity is directed towards saving life or enhancing it. Among the various factors which put life in danger are natural or human-created disasters. To save human life in such situations, it is imperative to evacuate people to safe places. Evacuation planning is studied using mathematical models expressing them as network problem as per requirements, e.g. maximum flow problem, lexicographic flow problem, quickest flow problem, earlies arrival flow problem, time dependent problem, contraflow problem, etc. (Dhamala, 2015).

Bus Evacuation Problem, BEP

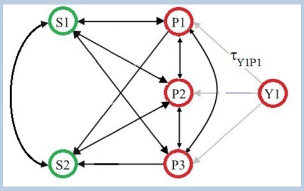


Figure 1: BEP network

Most of the evacuation problems studied so far address auto-based evacuation, or building evacuation (Hua, Ren, Cheng, & Ran, 2014; Bish, 2011). To address the need of the population which depends on the public vehicles, Bish(2011) devised the evacuation problem as a bus evacuation problem (BEP). The solution of BEP gives the routes to be followed by a given number of buses and number of evacuees to be assigned to particular bus so that the evacuation time is minimized. Although it is based on well-known vehicle routing problem (VRP), it differs from traditional VRP and other similar evacuation models in that it minimizes the time of the vehicle which takes the longest time for evacuation while most of the other models use minimum cost flow model which minimizes the total cost of evacuation. The fact that BEP surpasses the minimum cost problems in minimizing the evacuation time can be realized by the following ordinary observation

$$1 + 2 + 7 = 1 + 4 + 5$$

Although the sum is the same, the maximum of the constituents is smaller in the second.

Mathematical Model of BEP

The BEP is modeled as a network (N, A) where N, the set of nodes is the union of Y, the set of yards, P, the set of pick-up locations, and S, the set of shelters. The set of available buses with equal capacity Q is denoted by V which is the union of the set of buses available at an yard i, V_i's. D_j denotes demand (number of evacuees) at the pick-up node j, and C_i, the capacity of a shelter i. The travel cost along the arc (i, j) is denoted by τ_{ij}.The decision variables are x_{ij}^{mt}, b_j^{mt}. x_{ij}^{mt} is a binary variable which equals 1 if trip t of a bus m traverses arc (i, j), whereas b_j^{mt} is the number of evacuees from node j assigned to (or, if j is a shelter, released from) bus m after trip t.

$$\begin{aligned}
&\text{Minimize} \quad T_{evac} && && (1)\\
&\text{subject to: } T_{evac} &\geq\;& \sum_{(i,j)\in A}\sum_{t=1}^{T} \tau_{ij} x_{ij}^{mt}, \quad \forall m \in V && (2)\\
&\sum_{i:(i,j)\in A} x_{ij}^{mt} &=\;& \sum_{k:(j,k)\in A} x_{jk}^{m(t+1)}, \quad \forall j \in P, m \in V, t = 1, \dots, T-1 && (3)\\
&\sum_{i:(i,j)\in A} x_{ij}^{mt} &\geq\;& \sum_{k:(j,k)\in A} x_{jk}^{m(t+1)}, \quad \forall j \in S, m \in V, t = 1, \dots, T-1 && (4)\\
&\sum_{(i,j)\in A} x_{ij}^{mt} &\leq\;& 1, \quad \forall m \in V, t = 1 \dots, T && (5)\\
&x_{ij}^{m1} &=\;& 1, \quad \forall i \in Y, j : (i,j) \in A, m \in V_i && (6)\\
&x_{ij}^{mt} &=\;& 0, \quad \forall i \in Y, j : (i,j) \in A, m \in V, t = 2, \dots, T && (7)\\
&x_{ij}^{mT} &=\;& 0, \quad \forall j \in P, i : (i,j) \in A, m \in V && (8)\\
&b_j^{mt} &\leq\;& \sum_{(i,j)\in A} Q x_{ij}^{mt}, \quad \forall j \in N, m \in V, t = 1, \dots, T && (9)\\
&0 &\leq\;& \sum_{j\in P}\sum_{l=1}^{t} b_j^{ml} - \sum_{k\in S}\sum_{l=1}^{t} b_k^{ml} \leq Q, \quad \forall m \in V, t = 1, \dots, T && (10)\\
&\sum_{m\in V}\sum_{t=1}^{T} b_j^{mt} &\leq\;& C_j, \quad \forall j \in S && (11)\\
&\sum_{m\in V}\sum_{t=1}^{T} b_j^{mt} &=\;& D_j, \quad \forall j \in P && (12)\\
&\sum_{j\in P}\sum_{t=1}^{T} b_j^{mt} &=\;& \sum_{k\in S}\sum_{t=1}^{T} b_k^{mt}, \quad \forall m \in V && (13)\\
&b_j^{mt} &\geq\;& 0, \quad \forall (i,j) \in A, m \in V, t = 1, \dots, T && (14)
\end{aligned}$$

BEP Variants

Realizing the importance of BEP, Goerigk, Grün, & Heßler (2013) present a simplified version of it assuming the number of evacuees to be equal to the integer multiples of bus capacities, and considering all buses to be located at a single yard initially.

Goerigk and Grün(2014) adapted BEP model to situations when the number of evacuees is known only after some time the evacuation begins with aforementioned assumptions.

BEP with Sufficient Buses

Although BEP is crucial for planning evacuation for minimum evacuation time, it is difficult to solve, becuase buses move from pick-up nodes to shelter, between pick-up nodes, from shelter to pick-up nodes, and between shelters. To make the network structure simpler, the idea used by Hua et al. (2014) can be applied in BEP. Dividing an evacuation in several evacuation zones, they estimate the number of buses required at a pick-up location representing a zone by $\frac{P^i - aV^i_{auto}}{b}$ where V^i_{auto} is the number of auto-vehicles in the zone, a is the average no. of occupants in an auto-vehicle, and b is the capacity of a transit-vehicle.

In this way, if the number of buses made available in BEP becomes

$$\sum \lceil \frac{D_j}{Q} \rceil$$

and if the yards are close enough to the pick-up locations, the BEP network can be simplified to the one given in figure 2.

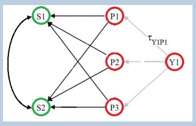


Figure 2: Simplified BEP network

Moreover, if the shelters are uncapacitated, the number of buses calculated, in this way, is optimal.

Heuristic Solution

To find a quick feasible solution to BEP with sufficient number of buses, the following heuristic (suggested by Bish(2011) with necessary modifications) can be used. The feasible solution can be improved using other optimization techniques.

Step1 Produce a list of buses sequenced in nondecreasing order of their total number of movements from a yard to a pickup location plus the movements from a pickup location to a shelter, and then further sequenced by the travel costs of the routes assigned to each bus, breaking ties arbitrarily.

Step2 Denote the first bus in the list as bus i. If bus i is at a pickup location, route it to the nearest shelter with sufficient remaining capacity, and if bus i is at a yard, route it to the nearest pickup location that has remaining evacuees. Once there, pick up as many evacuees as the vehicle capacity allows. Update the list of buses.

Step 3 Go to Step 1 and continue until all evacuees are in shelters.

Conclusion

If sufficient number of buses is available, the BEP network gets simplified reducing the movement of buses from shelter to pick-up nodes, and between pick-up nodes making it relatively easier to solve.

References

Bish, D.R. (2011). Planning for a bus-based evacuation. *OR Spectrum, 33*, 629-654.
Dhamala, T.N. (2015). A survey on models and algorithms for discrete evacuation planning network problems. *Journal of Industrial and Management Optimization, 11*(1), 265-289.
Goerigk, M., & Grün, B. (2014). A robust bus evacuation model with delayed scenario information. *OR Spectrum.*
Goerigk, M., Grün, B., & Heßler P. (2013). Branch and bound algorithms for the bus evacuation problem. *Computers & Operations Research, 40*,3010-3020.
Hua, J., Ren, G., Cheng, Y. & Ran, B. (2014). An integrated contraflow strategy for multimodal evacuation. *Mathematical problems in engineering.*

Acknowledgements

I am grateful to Prof. Dr. Tanka Nath Dhamala for continuous guidance, Dr. Urmila Pyakurel for technical support, and Society for Mathematical Biology for travel support.

DEPARTMENT *of* MATHEMATICS *and* STATISTICS
COLLEGE OF SCIENCE AND MATHEMATICS
MSU-ILIGAN INSTITUTE OF TECHNOLOGY

presents this

CERTIFICATE OF APPRECIATION

to

HARI NANDAN NATH

for presenting the paper entitled
"Meaningfulness of OR Models and Solution Strategies for Emergency Planning"
during its

MATHEMATICS AND STATISTICS RESEARCH COLLOQUIUM

held on 17 August 2016 at Lecture Hall B,
College of Science and Mathematics, MSU-Iligan Institute of Technology,
Iligan City, Philippines.

Given this 17th day of August, 2016.

[illegible] [illegible]

JULIUS V. BENITEZ, Ph.D.
CHAIRPERSON
Dept. of Mathematics and Statistics

MARK NOLAN P. CONFESOR, Ph.D.
DEAN
College of Science and Mathematics

National Conference on

Mathematics and Its Applications

(NCMA-2017)

January 11-13, 2017, Chitwan, Nepal

CERTIFICATE

This certificate is awarded to

Hari Nandan Nath

Tribhuvan University

for participating and presenting

a paper entitled

Identification of Optimal Pick-up Locations with their Demands in Evacuation Planning of Transit-dependent Population

in the National Conference on

Mathematics and Its Applications

organized by

Nepal Mathematical Society.

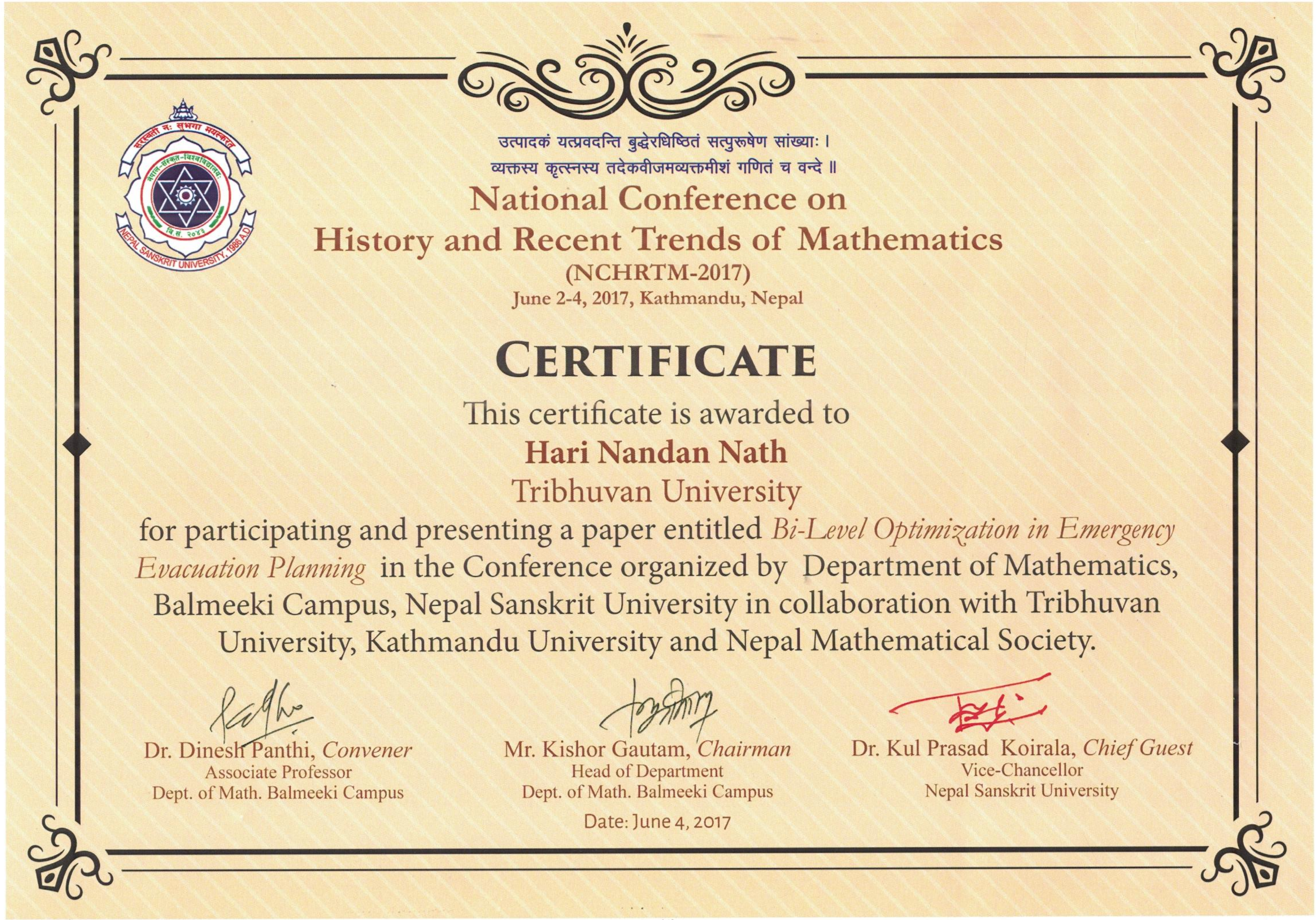
[illegible]

Prof. Dr. Tanka Nath Dhamala
President
Nepal Mathematical Society

[illegible]

Prof. Dr. Ishwari Prasad Dhakal
Vice-Chancellor
Agriculture and Forestry University, Nepal

Date: January 13, 2017



उत्पादकं यत्प्रवदन्ति बुद्धेरधिष्ठितं सत्पुरूषेण सांख्याः ।
व्यक्तस्य कृत्स्नस्य तदेकवीजमव्यक्तमीशं गणितं च वन्दे ॥

National Conference on History and Recent Trends of Mathematics

(NCHRTM-2017)

June 2-4, 2017, Kathmandu, Nepal

CERTIFICATE

This certificate is awarded to

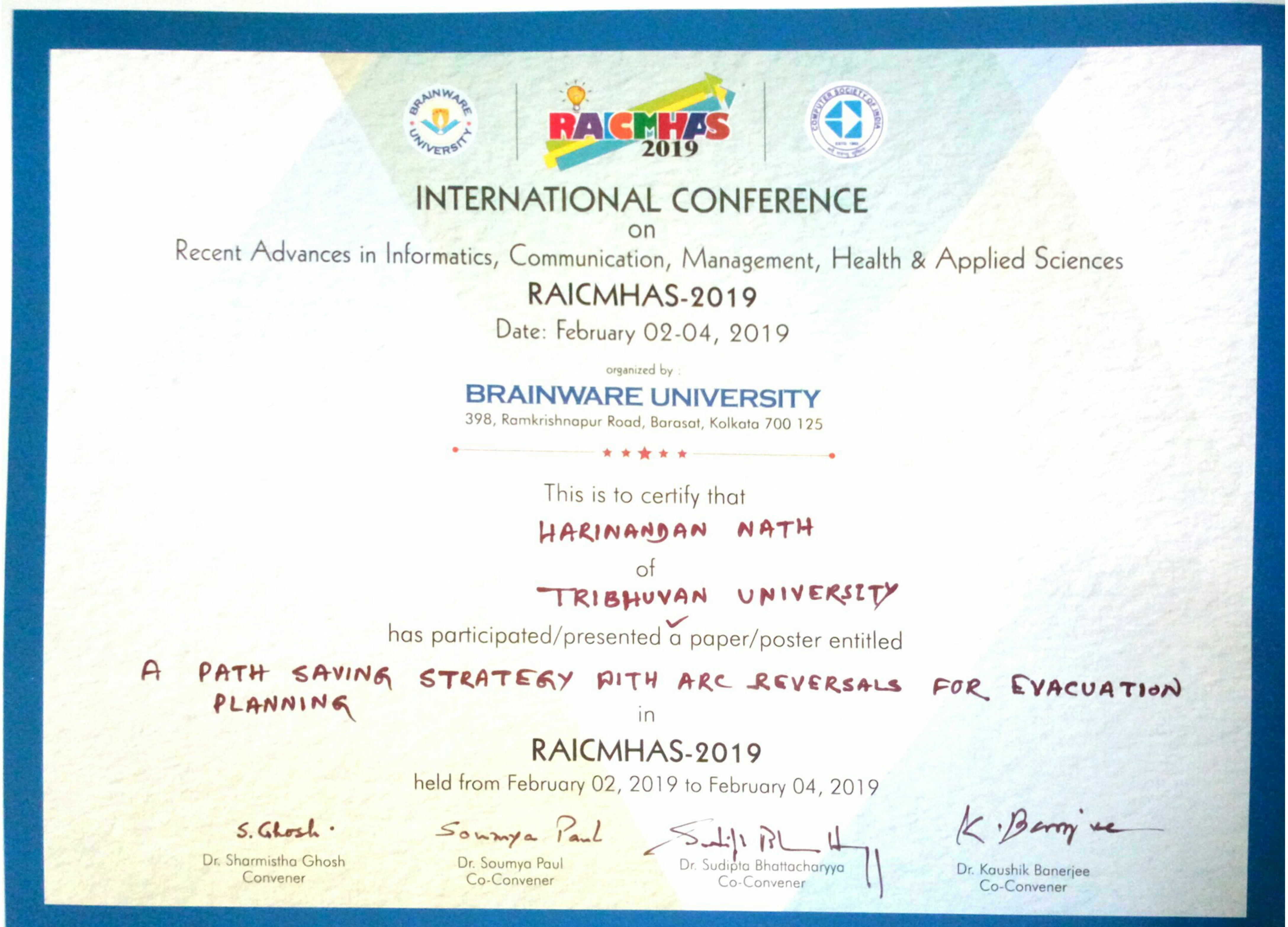
Hari Nandan Nath

Tribhuvan University

for participating and presenting a paper entitled *Bi-Level Optimization in Emergency Evacuation Planning* in the Conference organized by Department of Mathematics, Balmeeki Campus, Nepal Sanskrit University in collaboration with Tribhuvan University, Kathmandu University and Nepal Mathematical Society.

[illegible]	[illegible]	[illegible]
Dr. Dinesh Panthi, *Convener*	Mr. Kishor Gautam, *Chairman*	Dr. Kul Prasad Koirala, *Chief Guest*
Associate Professor	Head of Department	Vice-Chancellor
Dept. of Math. Balmeeki Campus	Dept. of Math. Balmeeki Campus	Nepal Sanskrit University

Date: June 4, 2017



INTERNATIONAL CONFERENCE
on
Recent Advances in Informatics, Communication, Management, Health & Applied Sciences

RAICMHAS-2019

Date: February 02-04, 2019

organized by :

BRAINWARE UNIVERSITY

398, Ramkrishnapur Road, Barasat, Kolkata 700 125

This is to certify that

HARINANDAN NATH

of

TRIBHUVAN UNIVERSITY

has participated/presented ✓ a paper/poster entitled

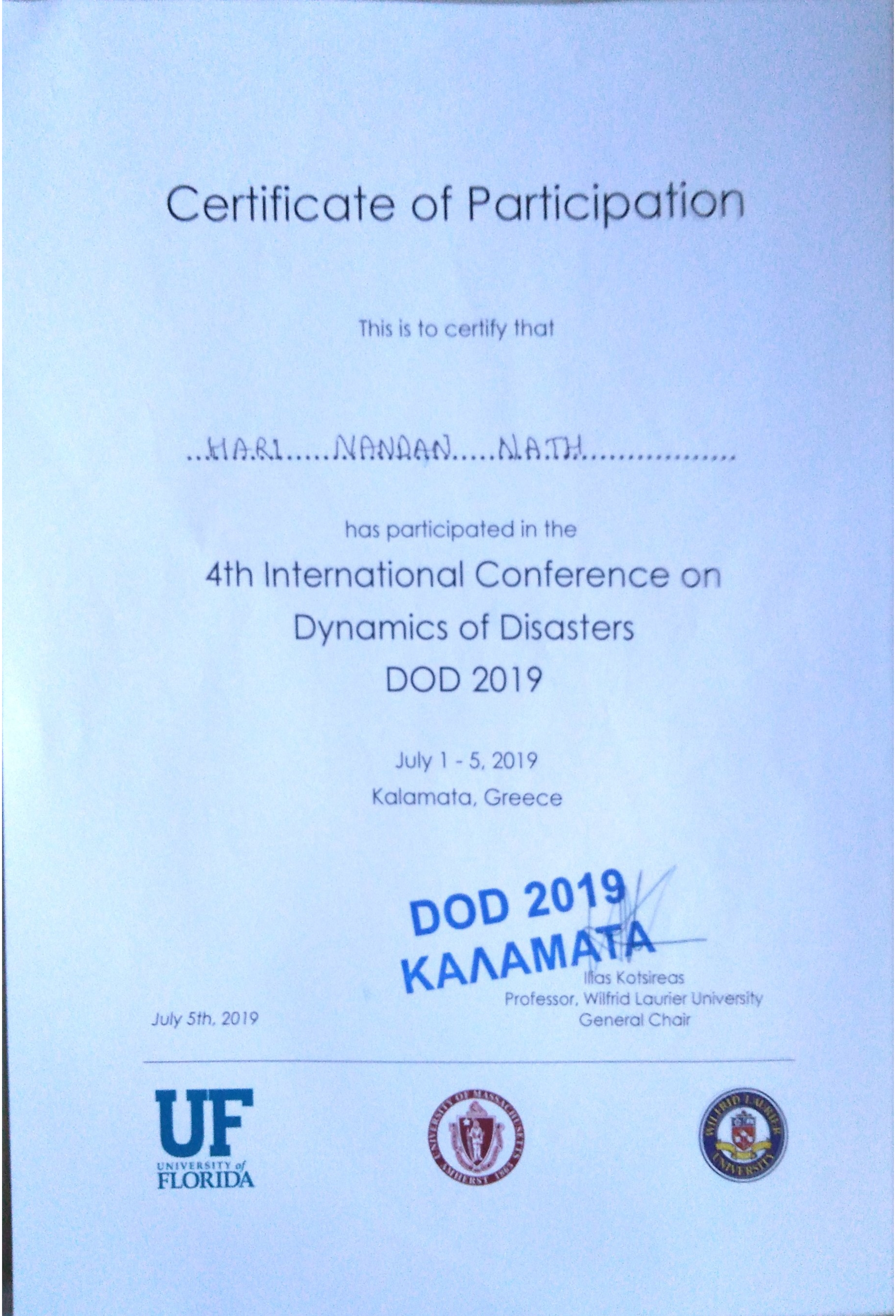
A PATH SAVING STRATEGY WITH ARC REVERSALS FOR EVACUATION PLANNING

in

RAICMHAS-2019

held from February 02, 2019 to February 04, 2019

S. Ghosh ·	Soumya Paul	[illegible]	[illegible]
Dr. Sharmistha Ghosh Convener	Dr. Soumya Paul Co-Convener	Dr. Sudipta Bhattacharyya Co-Convener	Dr. Kaushik Banerjee Co-Convener



Certificate of Participation

This is to certify that

..HARI.....NANDAN.....NATH..................

has participated in the

4th International Conference on
Dynamics of Disasters
DOD 2019

July 1 - 5, 2019
Kalamata, Greece

DOD 2019
ΚΑΛΑΜΑΤΑ

[illegible]

Ilias Kotsireas
Professor, Wilfrid Laurier University
General Chair

July 5th, 2019

UF
UNIVERSITY *of* FLORIDA

UNIVERSITY OF MASSACHUSETTS AMHERST 1863

WILFRID LAURIER UNIVERSITY

तमसो मा ज्योतिर्गमय

CSMCS 2020

INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCES- MODELLING, COMPUTING AND SOFT COMPUTING

CSMCS-2020

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

SEPTEMBER 10-12, 2020

CERTIFICATE

This is to certify that **Mr. Hari Nandan Nath**, **Central Department of Mathematics, Tribhuvan University, Kathmandu, Nepal**, has participated in the virtual web **"International Conference on Computational Sciences-Modelling, Computing and Soft Computing"** held online during **September 10-12, 2020** organized by the **Department of Mathematics** of **National Institute of Technology Calicut, Kerala**.

He /She has also presented the paper entitled **The Quickest FlowLoc Problem**.

[illegible]
Dr. Ashish Awasthi
Convener

[illegible]
Dr. Sunil Jacob John
Convener

[illegible]
Dr. Satyananda Panda
Chairperson

TEQIP - III TECHNICAL EDUCATION QUALITY IMPROVEMENT PROGRAMME - III Springer Wolfram Mathematica AIP